CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

# Intergenerational Interactive Artificial Neural Networks

ZKEIA ABDALLA ABDRHMAN JAZAM

*Department of Computer Engineering*

May , 2025

**CUKUROVA UNIVERSITY**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**

## PhD THESIS APPROVAL

---

# Intergenerational Interactive Artificial Neural Networks

---

ZKEIA ABDALLA ABDRHMAN JAZAM

***Department of Computer Engineering***

This Doctorate Thesis was evaluated by the following Jury Members on …/.../…… and was approved by unanimity/majority of votes.

| | | | |
|---|---|---|---|
| Jury | : Prof. Dr. Mutlu AVCI | (Advisor) | ……….. |
| | : Prof. Dr. Selma Ayse ÖZEL | | ……….. |
| | : Prof. Dr. Serdar YILDIRIM | | ……….. |
| | : Assoc. Prof. Dr. Ali İNAN | | ……….. |
| | : Assoc. Prof. Dr.Serkan KARTAL | | ……….. |

**This Thesis was written in the Department of Computer Engineering, Institute of Natural and Applied Sciences.**

**Thesis Number:**

**Prof. Dr. Sadık DİNÇER**
**Director**
**Institute of Natural and Applied Sciences**

**This work was supported by the XXXXX.**
**Project ID: XXX-XXXX-XXXX**

**İÇİNDEKİLER**

ABSTRACT ..... I
ÖZ ..... II
GENİŞLETİLMİŞ ÖZET ..... III
EXTENDED ABSTRACT ..... VI
ACKNOWLEDGEMENTS ..... IX
LIST OF TABLES ..... X
LIST OF FIGURES ..... XI
SYMBOLS AND ABBREVIATIONS ..... XIII
1. INTRODUCTION ..... 1
  1.1. Research Description ..... 3
  1.2. The Research Goal ..... 4
  1.3. Research Outline ..... 5
2. PRELIMINARY WORK ..... 7
3. MATERIAL AND METHOD ..... 25
  3.1 Datasets ..... 27
    3.1.1 MNIST dataset ..... 28
    3.1.2 Fashion-MNIST dataset ..... 28
    3.1.3 Cifer10 datasets ..... 28
    3.1.4 Bird dataset ..... 29
    3.1.5 STL-10 dataset ..... 29
    3.1.6 Cifer-100 datasets ..... 30
  3.2 Data Preprocessing (Image preprocessing) ..... 30
    3.2.1 The Normalization ..... 32
  3.3 Supervised learning ..... 33
  3.4 Unsupervised learning ..... 34
    3.4.1 Process of unsupervised learning ..... 34
  3.5 Classification Analysis ..... 35
  3.6 Hardware and Frameworks ..... 35
  3.7 The Software environment implementation in this thesis ..... 36
    3.7.1 PyTorch ..... 36
    3.7.2 TensorFllow ..... 36
    3.7.3 Keras ..... 38
  3.8 Ensemble Learning ..... 40
  3.9 Teacher-Student Network ..... 40
    3.9.1 Essential Elements ..... 41

3.9.2 Knowledge Distillation ........................................ 41
3.9.3 The general steps for knowledge distillation are as follows: ........................................ 41
3.10 Self-organizing map ........................................ 42
3.10.1 Self-Organization Mechanisms in SOMs ........................................ 43
3.10.2 The SOM Common Topologies ........................................ 47
3.11 Euclidean Distance ........................................ 50
3.12 Manhattan Distance ........................................ 50
3.13 Neighborhood Functions ........................................ 51
3.14 Deep learning ........................................ 52
3.14.1 Convolutional neural network ........................................ 55
3.14.1.1 Convolutional layer ........................................ 56
3.14.2 Convolutional Auto encoder neural network ........................................ 57
3.15 Deep convolutional Self-organizing map ........................................ 59
3.15.1 The Architecture ........................................ 60
3.15.2 Convolutional Layers ........................................ 60
3.15.3 SOM Layer ........................................ 60
3.15.4 Training ........................................ 60
3.15.5 Clustering and Visualization ........................................ 61
3.16 Differential convolutional neural networks ........................................ 61
3.17 Intergenerational Interaction Neural Networks ........................................ 64
3.18 The proposed Father-Son Network: Self-organizing map Differential Convolutional neural network ........................................ 65
3.18.1 The Proposed Method Architecture ........................................ 66
3.18.2 Training ........................................ 67
3.18.3 Classification and Visualization ........................................ 67
3.18.4 Algorithm2: the proposed neural network SOMdiffCNN Model Algorithm .......... 68
3.19 The Structured Models ........................................ 69
3.20 Evaluation Metrics ........................................ 73
3.20.1 Accuracy ........................................ 73
3.20.2 F1-Score ........................................ 73
3.20.3 Precision ........................................ 74
3.20.4 Recall ........................................ 74
4. RESULTS AND DISCUSSIONS ........................................ 77
4.1 The datasets ........................................ 77
4.2 The Experiments ........................................ 78
4.2.1 The initial set ........................................ 78
4.2.2 The Second series of Experiments ........................................ 79

4.2.3 The Third set of Experiments ........ 79
4.2.4 The Fourth experiment set ........ 79
4.2.5 The fifth set of experiments ........ 79
4.2.6 Experiment Sixth ........ 80
4.2.7 Experiment seven ........ 80
4.3 The Results ........ 80
4.3.1 Experiment 1 ........ 80
4.3.2 The experiment 2 ........ 80
4.3.3 Experiment 3 ........ 81
4.3.4 Experiment 4 ........ 81
4.3.5 Experiment 5 ........ 81
4.3.6 Experiment 6 ........ 81
4.3.7 Experiment 7 ........ 81
5. CONCLUSIONS ........ 99
REFERENCES ........ 101
CURRICULUM VITAE ........ 127

**CUKUROVA UNIVERSITY**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**

**PhD THESIS**

# Intergenerational Interactive Artificial Neural Networks

**ZKEIA ABDALLA ABDRHMAN JAZAM**

***Advisor: Prof. Dr. Mutlu AVCI***

***Department of Computer Engineering***

## ABSTRACT

Deep learning has become increasingly prevalent across diverse fields, driven by advanced learning techniques and complex network architectures such as transfer learning and teacher-student models. Transfer learning aims to achieve high performance in a target domain by leveraging knowledge from a related source domain. In contrast, teacher-student models distill knowledge from a large, complex teacher model to a smaller student model, allowing for reduced complexity without significantly compromising accuracy. Convolutional Neural Networks (CNNs), widely used in image recognition, play a key role in such models due to their efficiency and performance. However, due to their complexity, CNNs often demand substantial computational resources and long training times. This study introduces a novel hybrid neural network topology: Intergenerational Interaction Neural Networks **(**IINNs**)**. The proposed hybridization topology is called Intergenerational Interaction Neural Networks. The hypothesis behind this method is that "the presence of a guiding father model enables the son model to succeed quicker and better than the others". This philosophy can be extended by incorporating additional ancestors, such as grandfathers and great-grandfathers. Unlike traditional teacher-student models, IINNs employ a pre-trained ancestor model (father) that remains static during training but actively guides the learning of the Son model. Specifically, a Self-Organizing Map (SOM) acts as the pre-trained father, and a Differential Convolutional Neural Network (DiffCNN) functions as the son. The SOM's outputs are integrated into the DiffCNN's training, enhancing convergence speed and accuracy while reducing convolutional complexity. The proposed model was evaluated on six datasets: MNIST, FashionMNIST, Birds, STL10, CIFAR-100, and CIFAR-10. It achieved superior accuracy scores of 98.58%, 96.53%, 87.49%, 86.99%, 86.78%, and 81.65%, respectively, outperforming state-of-the-art CNN, DiffCNN, and Deep Convolutional SOM models. Moreover, the model demonstrated faster convergence up to 84%, reaching 85% accuracy on more complex datasets, such as CIFAR-10, Birds, and CIFAR-100, within the first 7 to 10 epochs. At the same time, it maintained strong performance across simpler datasets like FashionMNIST, where it reached 90% accuracy by the 7th epoch, resulting in a 74% faster convergence. These results underscore the effectiveness and versatility of IINNs in accelerating training, faster convergence, and improving performance across simple and complex datasets, making them suitable for applications in medical imaging, automotive systems, and real-time scenarios like autonomous driving.

***Keywords***: Deep learning; Convolutional neural networks; Self-organizing maps; Differential Convolutional neural networks and Image datasets.

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

# Kuşaklararası Etkileşimli Yapay Sinir Ağları

**Adı SOYADI**
**ZKEIA ABDALLABDRHMAN JAZAM**

***Danışman: Prof. Dr. Mutlu AVCI***

***Bilgisayar Mühendislik Anabilim Dalı***

## ÖZ

Derin öğrenme, aktarımlı öğrenme ve öğretmen-öğrenci modelleri gibi gelişmiş öğrenme teknikleri ve karmaşık ağ mimarilerinin etkisiyle çeşitli alanlarda giderek daha yaygın hale gelmiştir. Aktarımlı öğrenme, ilgili bir kaynak alandan elde edilen bilgiyi kullanarak hedef alanda yüksek performans sağlamayı amaçlar. Buna karşılık, öğretmen-öğrenci modelleri, büyük ve karmaşık bir öğretmen modelinden daha küçük bir öğrenci modele bilgi aktarımı yaparak, doğruluktan fazla ödün vermeden modelin karmaşıklığını azaltmayı sağlar. Görüntü tanıma alanında yaygın olarak kullanılan Konvolüsyonel Sinir Ağları (CNN'ler), verimlilikleri ve yüksek performansları nedeniyle bu tür modellerde kilit rol oynar. Ancak, karmaşık yapıları nedeniyle CNN'ler genellikle yüksek hesaplama kaynakları ve uzun eğitim süreleri gerektirir. Bu çalışma, yeni bir hibrit sinir ağı topolojisi olan Kuşaklararası Etkileşimli Sinir Ağları (Intergenerational Interaction Neural Networks IINNs)'nı tanıtmaktadır. Önerilen bu hibrit topoloji, Kuşaklararası Etkileşimli Sinir Ağları (IINNs) olarak adlandırılmaktadır. Bu yöntemin arkasındaki temel hipotez, *"yol gösterici bir baba modelin varlığı, oğul modelin diğerlerinden daha hızlı ve başarılı şekilde öğrenmesini sağlar"* şeklindedir. Bu felsefe, dede ve büyükbaba gibi ek ataların dahil edilmesiyle genişletilebilir. Geleneksel öğretmen-öğrenci modellerinden farklı olarak, IINNs mimarisi, eğitim sürecinde sabit kalan ancak oğul modelin öğrenmesini aktif şekilde yönlendiren, önceden eğitilmiş bir ata modeli (baba) kullanır. Bu yapıda, Kendini Örgütleyen Harita (Self-Organizing Map - SOM) önceden eğitilmiş baba model olarak görev yaparken, Farklılaştırılmış Konvolüsyonel Sinir Ağı (Differential CNN - DiffCNN) oğul modeli olarak işlev görür. SOM'un çıktıları, DiffCNN'in eğitim sürecine entegre edilerek yakınsama hızını ve doğruluğunu artırmakta, konvolüsyonel karmaşıklığı ise azaltmaktadır. Önerilen model, MNIST, FashionMNIST, Birds, STL10, CIFAR-100 ve CIFAR-10 olmak üzere altı veri kümesi üzerinde değerlendirilmiştir. Sırasıyla %98,58, %96,53, %87,49, %86,99, %86,78 ve %81,65 doğruluk skorları elde ederek; en güncel CNN, DiffCNN ve Derin Konvolüsyonel SOM modellerinden daha iyi performans göstermiştir. Ayrıca model, %84'e kadar daha hızlı yakınsama sergilemiş; CIFAR-10, Birds ve CIFAR-100 gibi daha karmaşık veri kümelerinde ilk 7 ila 10 epoch içinde %85 doğruluğa ulaşmıştır. Aynı zamanda, FashionMNIST gibi daha basit veri kümelerinde 7. epoch itibariyle %90 doğruluğa erişmiş ve %74 oranında daha hızlı yakınsama sağlamıştır. Bu sonuçlar, IINNs mimarisinin eğitim sürecini hızlandırmada, daha hızlı yakınsamayı sağlamada ve basit ya da karmaşık veri kümelerinde performansı artırmada etkili ve çok yönlü olduğunu göstermektedir. Bu yönleriyle, model tıbbi görüntüleme, otomotiv sistemleri ve otonom sürüş gibi gerçek zamanlı uygulamalar için uygun bir çözüm sunmaktadır.

**Anahtar Kelimeler:** Derin öğrenme; Konvolüsyonel sinir ağları; Öz-örgütlenen haritalar; Diferansiyel konvolüsyonel sinir ağları; Görüntü veri seti.

## GENİŞLETİLMİŞ ÖZET

Derin öğrenme, son yıllarda gösterdiği üstün performans ve çok çeşitli alanlardaki geniş uygulama yelpazesi sayesinde hızla önem kazanmıştır. Görüntü işleme, doğal dil işleme, büyük ölçekli veri analizi ve internet aramaları gibi alanlardan, makinelerin karmaşık verilerle nasıl öğrendiği ve etkileşim kurduğuna kadar birçok süreci köklü biçimde dönüştürmüştür. Derin öğrenme yöntemlerinin temelinde, insan beyninin yapısından ve işleyişinden esinlenen yapay sinir ağları yer almaktadır. Bu ağlar, genellikle birden fazla katmandan oluşur; bu nedenle "derin" öğrenme olarak adlandırılır. Katmanlar, girdileri giderek daha soyut ve bilgilendirici özelliklere dönüştürerek verinin hiyerarşik temsillerini öğrenmelerine olanak tanır. Bu çok katmanlı yapı, artan hesaplama gücü ve büyük veri kümelerine erişimin kolaylaşmasıyla birleştiğinde, derin öğrenmenin farklı gerçek dünya uygulamalarında yüksek etkililik ve esneklik göstermesine önemli ölçüde katkı sağlamıştır.Bu yaklaşımlar genellikle iki ana türe ayrılır: Derin İnanç Ağları (DBNs) (Le Roux & Bengio, 2008) ve Konvolüsyonel Sinir Ağları (CNNs) (Yuda et al., 2020). DBNs, temel, olasılıksal, katmanlı bir ağ modeliyken, CNNs, görüntü tanıma gibi yüksek performans gerektiren görevler için tasarlanmış, özel konvolüsyonel katmanlara sahip çok katmanlı ağlardır. CNN'lerdeki konvolüsyonel katmanlar, hiyerarşik özellikleri çıkarmak için maksimum havuzlama işlemlerini kullanır (O'Shea & Nash, 2015). Bu derin katmanlı ağların eğitimini optimize etmek için statik olarak düzenlenmiş başlangıç veya hafif öğrenme teknikleri gibi yöntemler önerilmiştir (Acharya et al., 2017). Ancak, daha fazla ilerleme sağlamak için diğer sinir ağı mimarilerinin kullanılması gerekmektedir. Görüntü işleme ve desen tanıma sistemlerinin çoğu, görselleştirme için bir hiyerarşi eğitmeye vurgu yapmaktadır (Danuser, 2011).

Başarılarına rağmen, Evrişimli Sinir Ağları (Convolutional Neural Networks - CNN), genellikle büyük ağ boyutlarına sahip olmaları nedeniyle yüksek hesaplama maliyetleri ve uzun eğitim süreleri gibi sorunlarla karşı karşıya kalmaktadır. Bu tür sınırlılıklar, özellikle kaynak kısıtlı ortamlarda veya gerçek zamanlı uygulamalarda bu modellerin kullanımını zorlaştırabilmektedir. Ayrıca, CNN mimarilerinin büyük çoğunluğu denetimli öğrenmeye odaklanmakta ve büyük etiketli veri kümelerine olan bağımlılıklarını sürdürmektedir. Buna karşın, denetimsiz öğrenme tekniklerinin araştırılması görece olarak daha az ilgi görmüştür. Etiketli veriye ihtiyaç duymayan denetimsiz yöntemler, veri etiketleme maliyetlerini azaltma ve modelin genelleme yeteneğini artırma açısından önemli bir potansiyele sahip olmasına rağmen, mevcut CNN geliştirme süreçlerinde yeterince değerlendirilememektedir.

K-means (Ahmad & Dey, 2007), PCANet (Chan et al., 2015), ScatNet (Feng et al., 2021) ve SOMNet (Hankins et al., 2018) gibi denetimsiz yöntemler, derin öğrenmeye uygulanmıştır, ancak denetimli öğrenme tekniklerine kıyasla daha az kullanılmaktadır.

Transfer öğrenimi, derin sinir ağlarının eğitim sürecinde karşılaşılan veri yetersizliği ve yüksek hesaplama maliyeti gibi zorlukları aşmak amacıyla etkili bir teknik olarak öne çıkmıştır. Bu

yaklaşım, daha önce büyük ve ilişkili veri kümeleri üzerinde eğitilmiş bir modelin, daha küçük ölçekli ve alanına özgü görevler için yeniden kullanılmasına olanak tanır. Böylece, hem etiketli veri ihtiyacı önemli ölçüde azalır hem de modelin eğitim süreci hızlanır. Transfer öğrenimi, özellikle bilgisayarlı görü ve doğal dil işleme gibi büyük veri gerektiren alanlarda yaygın bir şekilde kullanılmakta; modellerin daha hızlı öğrenmesini, daha yüksek doğrulukla sonuç üretmesini ve genelleme kapasitesinin artmasını sağlamaktadır. Bu yönüyle transfer öğrenimi, derin öğrenme modellerinin daha esnek ve verimli bir şekilde farklı problemlere uyarlanmasına önemli katkılar sunmaktadır (Pan & Yang, 2010). Transfer öğrenimi içinde bir yaklaşım olan öğretmen-öğrenci ağı, bilgi damıtma kullanarak karmaşık bir öğretmen modelinden daha basit bir öğrenci modeline bilgi aktarır ve kaynak kısıtlı ortamlarda yüksek performanslı modellerden faydalanılmasını sağlar (Hinton, Vinyals & Dean, 2015).

Transfer öğrenimi ve öğretmen-öğrenci çerçeveleri, verimlilik ve ölçeklenebilirliğe vurgu yaparken, genellikle iş birliğine dayalı, nesiller arası öğrenme paradigmalarının potansiyelini göz ardı etmektedir. İnsan ve doğal sistemlerdeki nesiller arası bilgi aktarımından ilham alan bu çalışmada, ardışık nesiller arasındaki ilişkilere dayalı yeni bir sinir ağı hibridizasyon yaklaşımı tanıtılmıştır. Önerilen hibridizasyon topolojisine Nesiller Arası Etkileşim Sinir Ağları (IINNs) adı verilmiştir. Bu yöntemin hipotezi, "bir yönlendirici baba modelin varlığının, oğul modelin diğerlerinden daha hızlı ve daha iyi başarı göstermesini sağladığıdır." Bu felsefe, büyükbaba ve büyük büyükbaba gibi ek ataları içerecek şekilde genişletilebilir. Öğretmen-öğrenci modelinden farklı olarak, önerilen yaklaşımda önceden eğitilmiş baba veya ata modeller, oğul model ile birlikte çalışır. Oğul sinir ağının hem eğitim hem de uygulama aşamalarında baba veya ata bölümlerinin ağırlık değerleri güncellenmez.Önerilen yaklaşım, baba olarak bir Özyönlendirmeli Harita (SOM) ve oğul olarak bir Diferansiyel Konvolüsyonel Sinir Ağı (DiffCNN) kullanılarak uygulanmıştır. Bu yapılandırmada SOM, bağımsız olarak eğitilerek yönlendirici baba işlevi görür. Eğitim sırasında modelin her adımında, SOM'dan kodlanmış girişler kullanılarak SOM'nin çıktıları tam bağlantılı katmanlara geçmeden önce düzleştirilerek görüntülerin kodlanmış çıktılarıyla birleştirilmiştir. Bu adaptasyon, konvolüsyonel katmanların karmaşıklığını azaltırken yakınsama hızını ve genel performansı artırmıştır.

Bu makalenin birincil amacı, derin öğrenme modellerinin performansını, yakınsama hızını ve hesaplama verimliliğini artırmak için yeni bir hibrit sinir ağı olan Nesiller Arası Etkileşim Sinir Ağı (SOMdiffCNN) mimarisini önermek ve değerlendirmektir. Bir yönlendirici baba modelin varlığı, oğul modelin daha hızlı ve daha başarılı bir şekilde öğrenmesini sağlarken, karmaşıklığı azaltır ve birden fazla görüntü veri setinde yüksek doğruluk elde eder. Makalenin bir diğer amacı, önerilen modelin performansını, DCSOM, DiffCNN ve CNN gibi mevcut en iyi yöntemlerle karşılaştırarak etkinliğini göstermektir. Bu çalışmayla, geleneksel hiyerarşik öğrenme yöntemleri ile daha iş birliğine dayalı, nesiller arası bir yaklaşım arasında köprü kurmayı, derin öğrenmenin kavramsal çerçevesini ve pratik uygulamalarını ilerletmeyi hedeflemektedir.

Önerilen model, MNIST, Fashion_MNIST, Birds, STL10, CIFAR-100 ve CIFAR-10 gibi altı yaygın kullanılan görüntü veri seti üzerinde test edilmiştir. Sonuç olarak sırasıyla %98.58, %96.53, %87.49, %86.99, %86.78 ve %81.65 sınıflandırma doğruluk değerlerine ulaşılmıştır. Ayrıca model, ilk 10 eğitim epoch'u içinde önemli performans iyileştirmeleri göstermiştir. Deneysel sonuçlar, eğitim ve test doğruluğunda önemli iyileştirmeler olduğunu göstermektedir. MNIST veri seti için model, eğitim doğruluğunda %10.62 ve test doğruluğunda %6.53 iyileşme sağlamıştır. Benzer şekilde, Fashion-MNIST, CIFAR-10 ve CIFAR-100 veri setlerinde model, sırasıyla %23.72 / %24.16, %20.48 / %9.42 ve %62.48 / %50.92 doğruluk iyileştirmeleri kaydetmiştir. Ayrıca, Birds ve STL-10 veri setlerinde de %26.92 / %31.14 ve %10.21 / %15.03 doğruluk artışları gözlemlenmiştir.

Önerilen model, ilk 7 ila 10 epoch içinde %85 doğruluğa ulaşarak %84'e varan hızlı yakınsama sağlamıştır. FashionMNIST gibi daha basit veri setlerinde 7. epoch'da %90 doğruluğa ulaşarak %74 daha hızlı yakınsama sağlamıştır. Sonuçlar, SOMdiffCNN'in mevcut yöntemlerden daha üstün olduğunu ve tüm veri setlerinde son teknoloji doğruluk elde ettiğini göstermektedir.bir öğrenme paradigmasını teşvik ettiği yönündeki hipotezi doğrulamaktadır. Kuşaklar arası yaklaşım, yalnızca hibrit sinir ağlarına yönelik teorik anlayışı geliştirmekle kalmamakta, aynı zamanda zorlu sınıflandırma görevlerinde model performansını artırmak için pratik bir çözüm de sunmaktadır. Bu çalışma, denetimli ve denetimsiz öğrenme ilkelerini birleştiren yenilikçi bir hibrit mimari sunarak derin öğrenme alanına katkı sağlamaktadır. Baba-Oğul Ağ Paradigması, kuşaklar arası bilgi aktarımının daha verimli ve uyarlanabilir sinir ağlarının inşasında ne denli önemli olduğunu vurgulamaktadır. Gelecekte yapılacak araştırmalar, bu çerçevenin ölçeklenebilirliğini, doğal dil işleme veya zaman serisi verileri gibi diğer alanlardaki uygulanabilirliğini, daha derin atasal yapıları ve ilave hiyerarşik kuşak katmanlarının entegrasyonunu inceleyerek performansı daha da artırma potansiyelini araştırabilir.

Geliştirilen topoloji; tıbbi görüntü tanıma, otomotiv sektöründe görüntü tanıma ve otomatik sürüş sistemleri gibi gerçek zamanlı görüntü tanıma görevlerinde daha yüksek performans avantajları ve daha hızlı öğrenme hızı ile uygulanabilir niteliktedir. Bu çalışma aracılığıyla, geleneksel hiyerarşik öğrenme yöntemleri ile daha iş birliğine dayalı, kuşaklar arası yaklaşım arasında bir köprü kurmak, hem kavramsal çerçeveyi hem de derin öğrenmenin pratik uygulamalarını ileriye taşımak amaçlanmaktadır. Bu çalışma, yapay zekânın gelişiminde biyolojik esinli metodolojilerin potansiyelini vurgulayarak, yeni nesil sinir ağı mimarilerinin tasarımı ve uygulanması için umut verici bir temel sunmaktadır. Özellikle canlı sistemlerde gözlemlenen kuşaklar arası bilgi aktarımı, adaptasyon ve iş birliği mekanizmalarından ilham alan bu yaklaşım, yapay öğrenme süreçlerine yeni bir boyut kazandırmaktadır. Bu yönüyle çalışma, hem biyolojik sistemlerin işleyişine dayalı yapay öğrenme yaklaşımlarının etkinliğini ortaya koymakta hem de gelecekteki yapay zekâ araştırmaları için yeni açılımlar sunmaktadır.

## EXTENDED ABSTRACT

Deep learning has rapidly gained prominence in recent years due to its remarkable performance and wide-ranging applicability across numerous domains. From image processing and natural language processing to large-scale data analysis and internet search, deep learning has revolutionized how machines learn and interact with complex data. At the heart of deep learning methodologies lies the artificial neural network, a computational model inspired by the structure and function of the human brain. These networks are typically composed of multiple layers hence the term "deep" learning, which allow them to learn hierarchical representations of data. Each layer transforms the input data into increasingly abstract and informative features, enabling the network to capture intricate patterns and relationships. This multilayered structure, along with advancements in computational power and the availability of large datasets, has significantly contributed to the effectiveness and versatility of deep learning across various real-world applications.These approaches are often categorized into two primary types: deep belief networks (DBNs) (Le Roux & Bengio, 2008) and convolutional neural networks (CNNs) (Yuda et al., 2020). DBNs are a fundamental, probabilistic, layered network model, while CNNs are multi-layered networks with specialized convolutional layers designed for high-performance tasks like image recognition. The convolutional layers in CNNs employ max pooling operations to extract hierarchical features (O'Shea & Nash, 2015). Methods like statically organized initialization or lightweight learning techniques have been proposed to optimize the training of such deeply layered networks (Acharya et al., 2017). And the other neural network architectures must be used to make more progress. Several computer vision and pattern recognition systems emphasize training a hierarchy of features for visualization (Danuser, 2011).

Despite their success, Convolutional Neural Networks (CNNs) often suffer from large network sizes, leading to high computational costs and prolonged training times. These limitations can hinder their deployment in resource-constrained environments or real-time applications. Moreover, while most CNN architectures focus on supervised learning, relying heavily on large labeled datasets, there has been comparatively less emphasis on exploring unsupervised learning techniques. Unsupervised methods, which do not require labeled data, hold significant potential for reducing data annotation costs and improving model generalization, yet remain underutilized in mainstream CNN development.Unsupervised methods like K-means (Ahmad & Dey, 2007), PCANet (Chan et al., 2015), ScatNet (Feng et al., 2021), and SOMNet (Hankins et al., 2018), have been applied to deep learning, they remain underutilized compared to supervised learning techniques.

Transfer learning has emerged as an effective technique to address the challenges of training deep neural networks. It allows models trained on large, related datasets to be adapted for smaller, domain-specific tasks, reducing the need for extensive computational resources and

labeled data. This method has found wide application in fields such as computer vision and natural language processing, enabling models to learn faster and achieve better performance (Pan & Yang, 2010). One approach within transfer learning, the teacher-student network, uses knowledge distillation to transfer knowledge from a complex teacher model to a simpler student model, enabling resource-constrained environments to benefit from high-performance models (Hinton, Vinyals, & Dean, 2015).

While transfer learning and teacher-student frameworks emphasize efficiency and scalability, they often overlook the potential of collaborative, intergenerational learning paradigms. Drawing inspiration from the intergenerational transmission of knowledge in human and natural systems.

In this work, a novel type of neural network hybridization approach based on the relationships among sequential generations is introduced. The proposed hybridization topology is called intergenerational interaction neural networks (IINNs). The hypothesis behind this method is that "the presence of a guiding father model enables the son model to succeed quicker and better than the others". This philosophy can be extended by including incorporating additional ancestors, such as grandfathers and great-grandfathers. Unlike the teacher-student model, the proposed approach involves pre-trained father or ancestor models working together with the son model. In both in training and application phases of the son neural network, father or ancestor parts do not update their weight values. The proposed approach was implemented by using a Self-Organizing Map (SOM) as the father and a Differential Convolutional Neural Network (DiffCNN) as the son. In this configuration, the SOM serves as the guiding father, independently pre-trained before connecting the son. The proposed NN by using the trained SOM during each training step of the model, the encoded inputs from the SOM were used, and the flattened output was concatenated with the encoded images before being passed through the fully connected layers. This adaptation reduced the complexity of the convolutional layers while improving convergence speed and overall performance.

The primary aim of this PhD Research is to propose and evaluate a novel hybrid neural network the Intergenerational Interaction Neural Network SOMdiffCNN architecture, to enhance the performance, convergence speed, and computational efficiency of deep learning models. By the presence of a guiding father model enables the son model to succeed quicker and better than the others reducing complexity, and achieving high accuracy on multiple image datasets. The work also aims to demonstrate the effectiveness of this approach by comparing the proposed model's performance with existing state-of-the-art methods, such as DCSOM, DiffCNN, and CNN, across various benchmark datasets. Through this work, the thesis seeks to advance the field of deep learning by introducing a new, efficient approach to neural network training and model deployment.

The proposed model was tested on six widely used image datasets: MNIST, Fashion_MNIST, Birds, STL10, CIFAR-100, and CIFAR-10. Resulting in classification accuracy values of 98.58%, 96.53%, 87.49%, 86.99%, 86.78%, and 81.65% respectively. Moreover, the model demonstrated significant performance improvements within the first 10 training epochs. Experimental results indicate notable enhancements in both training and testing accuracy. For the MNIST dataset, the model improved 10.62% in training accuracy and 6.53% in testing accuracy. Similarly, for the Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets, the model recorded accuracy improvements of 23.72% / 24.16%, 20.48% / 9.42%, and 62.48% / 50.92%, respectively. Additionally, notable performance gains were observed on the Birds and STL-10 datasets, with accuracy improvements of 26.92% / 31.14% and 10.21% / 15.03%, respectively. The proposed model achieves faster convergence and significant performance improvements across multiple datasets within the first 10 training epochs. The experimental results underscore the model's ability to achieve faster convergence up to 84%, reaching 85% accuracy on more complex datasets, such as CIFAR-10, Birds, and CIFAR-100 within the first 7 to 10 epochs. Maintaining strong performance across simpler datasets like FashionMNIST, where it reached 90% accuracy by the 7th epoch, resulting in a 74% faster convergence. The results demonstrate that the SOMdiffCNN outperforms existing methods such as DCSOM, DiffCNN, and CNN, achieving state-of-the-art accuracy on all datasets.

The findings validate the hypothesis that leveraging generational interactions within neural networks fosters a more efficient and effective learning paradigm. The intergenerational approach not only advances the theoretical understanding of hybrid neural networks but also offers a practical solution for enhancing model performance on challenging classification tasks.

This work contributes to the growing field of deep learning by presenting an innovative hybrid architecture that combines unsupervised and supervised learning principles. The Father-Son Network paradigm emphasizes the importance of leveraging intergenerational knowledge transfer to build more efficient and adaptive neural networks. Future research may explore the scalability of this framework, its applicability to other domains, such as natural language processing or time-series data, investigating the inclusion of deeper ancestral structures and the integration of additional hierarchical generational layers to further enhance performance. The developed topology is applicable to medical image recognition, image recognition in the automotive industry with better performance advantages, and real-time image recognition tasks such as automatic driving systems with much faster learning speed.

By introducing this work, the aim is to bridge the gap between traditional hierarchical learning methods and a more collaborative, intergenerational approach, advancing both the conceptual framework and practical applications of deep learning. This study sets a promising foundation for next-generation neural network architectures, highlighting the potential of biologically inspired methodologies in advancing artificial intelligence.

# ACKNOWLEDGEMENTS

First and foremost, I would like to express my heartfelt gratitude to my supervisor, **Prof.Dr. Mutlu AVCI**, for their unwavering guidance, encouragement, and insightful feedback throughout this research journey. Their expertise and mentorship have been invaluable in shaping this work and my growth as a researcher, He was the source of inspiration and a supportive father to me. I would also like to extend my sincere appreciation to the members of my supervisory committee, **Prof.Dr. Selma Ayse ÖZEL** and **Prof.Dr. Serdar YILDIRIM**, for their constructive suggestions and support, which significantly enriched the quality of this thesis.

I am deeply thankful to the faculty and staff of the Department of **Computer Engineering** and Department of **Biomedical Engineering** at **Çukurova University,** whose expertise and resources provided a robust foundation for my research. Their dedication and support created an environment that enabled me to thrive academically and personally.

I extend my heartfelt gratitude to **Asst. Prof. Mehmet SARIGÜL** for his invaluable guidance and unwavering support throughout this journey. His encouragement and insights have greatly contributed to the development of this work.

I am also profoundly thankful to the **Ministry of High Education in Sudan** and **Al-Geneina University** for their support and for laying the initial foundation of my academic journey, which enabled me to pursue advanced studies abroad.

A special thanks **to my family** for their unconditional love, patience, and encouragement, which have been a constant source of strength and inspiration. To **my friends** thank you for being my extended family, offering laughter, companionship, and support during the highs and lows of this journey.

I am profoundly grateful to the **YÖK Scholarship Program in Türkiye**, which afforded me the incredible opportunity to pursue my Ph.D. in this beautiful country. Your financial and academic support has been pivotal to my success, and I am deeply appreciative of the trust and investment you placed in me.

**Lastly,** thank you to all who have contributed to my academic and personal journey in ways both big and small. This thesis is as much a testament to your support as it is to my efforts.

## LIST OF TABLES

Table.2.1. Shows a Summary of the related work. 17
Table 3.1. The structured representation of the SOMdiffCNN Model 69
Table 3.2 The detailed structure of the DiffCNN model 70
Table 3.3 The Implemented CNN Structure. 71
Table 3.4. The Architecture of Deep Convolutional Self-Organizing Map 73
Table 4.1 Shows the dataset evaluation 78
Table 4.2 Shown the hyper-parameter settings for the proposed model used in all experiments 78
Table 4.3 The accuracies of our proposed SOMdiffCNN comparisons on six different image datasets are presented. 82
Table 4.4 The accuracy of our proposed method comparisons with DiffCNN, CNN, and 89
DCSOM on MNIST datasets are presented. 89
Table 4.5 Displays the accuracies of our proposed method compared to DiffCNN, CNN, and DCSOM on CIFAR-10 datasets. 89
Table 4.6 Shown the accuracies of our proposed approach comparisons with DiffCNN,CNN, and DCSOM on Birds datasets. 89
Table 4.7 Presentations the accuracies of our proposed method compared to DiffCNN,CNN, And DCSOM on the STL10 datasets. 90
Table 4.8 Confirmations the accuracies of our suggested Method comparisons with DiffCNN, CNN, and DCSOM on Fashion-mnist datasets are displayed. 90
Table 4.9 Displays the accuracies of our suggested method compared with DiffCNN, CNN, and DCSOM on Cifar100 datasets. 90
Table 4.10 Optimized parameters for the models. 98

**Şekil tablosu öğesi bulunamadı.**

## LIST OF FIGURES

Figure 1.1 Demonstration of general research methodology progress .......................................... 6
Figure 3.1 Shows the Research Material and Methodology Overview ....................................... 26
Figure 3.2. Shows MNIST datasets. The image is taken from .................................................... 28
Figure 3.3. Shows Fashion-MNIST datasets the image is taken from......................................... 28
Figure.3.4. Shows Cifer-10 datasets the image is taken from...................................................... 29
Figure.3.5 Shows Bird datasets. The image is taken from.......................................................... 29
Figure .3.6 Shows STL-10 datasets. The image is taken from ..................................................... 30
Figure 3.7 Shows Cifer-100 datasets. The image is taken from .................................................. 30
Figure 3.9 Shows the chart of the unsupervised learning model ................................................. 34
Figure 3.10. Shows Teacher-Student Network Structure.............................................................. 41
Figure 3.11 Shows the Gaussian neighborhood function before applying .................................... 45
Figure.3.12. Shows neuron neighborhoods and the most common SOM grids.............................. 48
Figure.3.13. Shows a 1-D SOM network, .................................................................................... 48
Figure 3.14. Shows a 2-D SOM network, .................................................................................... 48
Figure.3.15 Shows the architecture of the SOM network and the connections between the input and output function regions .......................................................................................... 48
Figure.3.16. Shows the difference between Manhattan Distance and Euclidean Distance ............. 50
Figure 3.17 Illustrates the progress of Deep Learning Models...................................................... 55
Figure 3.18 A Convolutional Neural Network Architecture including.......................................... 56
Figure 3.19 Shows Convolutional Auto-encoders. ....................................................................... 59
Figure 3.20 Showing DCSOM neural network architecture Diagram........................................... 60
Figure 3.21 Shows the available predefined filters for Diff-CNN................................................. 62
Figure 3.22 Shows how the differential feature maps are determined........................................... 63
Figure 3.23 Shows Intergenerational Interaction Neural Network Architecture ............................ 65
Figure.3.24 Showing the proposed neural network architecture Diagram..................................... 66
Figure 3.25 Shows the father-son Network Structure................................................................... 68
Figure 4.1 SOMdiffCNN Model Accuracies for Birds Dataset.................................................... 83
Figure 4.2 SOMdiffCNN Model Training Loss for Birds dataset. ............................................... 83
Figure 4.3. SOMdiffCNN Model Accuracies for STL10 Dataset. ................................................ 84
Figure 4.4 SOMdiffCNN Model Accuracies for STL10 Datasets.................................................. 84
Figure 4.5 SOMdiffCNN Model Accuracies for Fashion-mnist Dataset...................................... 85
Figure 4.6 SOMdiffCNN Model Training Loss for Fashion-mnist datasets. ................................ 85
Figure 4.7 SOMdiffCNN Model Accuracies for Cifar100 dataset. .............................................. 86
Figure 4.8 SOMdiffCNN Model Training Loss for Cifar100 Dataset........................................... 86
Figure 4.9 SOMdiffCNN Model Accuracies for Cifar10 Dataset. ............................................... 87

Figure 4.10  SOMdiffCNN Model  Training Loss  for Cifar10 Dataset. ........................................ 87
Figure 4.11  SOMdiffCNN Model Accuracies for MNIST Dataset. ............................................. 88
Figure 4.12  SOMdiffCNN Model Training Loss  for MNIST Dataset. ........................................ 88
Figure 4.13  Accuracy Comparison Results for Fashion-mnist Datasets. ..................................... 91
Figure 4.14  Loss Comparison Results for Fashion-mnist Datasets. ............................................. 91
Figure 4.15  Accuracy  Comparison Results for MNIST Datasets. ............................................... 92
Figure 4.16  Loss Comparison Results for MNIST Datasets ......................................................... 92
Figure 4.17  Accuracy Comparison Results for Cifar10 Datasets. ................................................ 93
Figure 4.18  Loss Comparison Results for Cifar10 Datasets. ........................................................ 93
Figure 4.19  Accuracy Comparison  Results for Birds Datasets. .................................................... 94
Figure 4.20  Loss Comparison Results for Birds dataset. .............................................................. 94
Figure 4.21. Accuracy Comparison Results for STL10 Datasets. .................................................. 95
Figure 4.22. loss Comparison Results for STL10 Datasets. ........................................................... 95
Figure 4.23. Accuracy Comparison Results for Cifar100 Datasets. ............................................... 96
Figure 4.24. loss Comparison Results for Cifar100 Datasets. ........................................................ 96

## SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| SOM | : Self-organizing map |
| CSOM | : Convolutional Self-organizing map |
| DCSOM | : Deep Convolutional Self-organizing map |
| Diff-CNN | : Differential Convolutional Neural Network |
| Diff-CSOM | : Differential Convolutional Self-organizing map |
| BP | : Back Propagation |
| AE | : Auto-encoder |
| DCAE | : Deep Convolutional Aotu encoder |
| SVM | : Support Victor Machine |
| 2D | : Two-Dimensional |
| DCNN | : Deep Convolutional Neural Network |
| ACS | : American Cancer Society |
| Adam | : Adaptive moment estimation |
| AI | : Artificial Intelligence |
| ANN | : Artificial Neural Network |
| CNN | : Convolutional Neural Network |
| Caps Net | : Capsule Neural Network |
| MDS | : Multi-Dimensional Scaling |
| ConvNet | : Convolutional Neural Network |
| DBM | : Deep Boltzmann Machine |
| SOMNet | : Self-Organizing Map Neural Network |
| DBN | : Deep Belief Network |
| BMU | : Best Match Unit |
| PCA | : Principle Component Analysis |
| DL | : Deep Learning |
| RGB | : Red Green Blue |
| DNN | : Deep Neural Network |
| LSTM | : Long-Short-Term Memories |
| EL | : Ensemble Learning |
| GPU | : Graphical Processing Unit |
| GRU | : Gated Recurrent Unit |
| HPC | : High-Performance Computing |
| ML | : Machine Learning |
| DNA | : Deoxyribonucleic acid |
| NLP | : Natural Language Processing |

|  |  |
|---|---|
| RBM | : Restricted Boltzmann Machine |
| ReLU | : Rectified Linear Unit |
| RNN | : Recurrent Neural Network |
| UL | : Unsupervised Learning |
| UAE | : Unsupervised Auto encoder |
| SGD | : Stochastic Gradient Descent |
| GAN | : Generative adversarial Network |
| ResNet | : Residual Networks |
| CSAE | : Classification Supervised Auto encoder |
| DDoS | : Distributed Denial of Service attack |
| MSE | : Mean Squared Error |
| BIRCH | : Balanced Iterative Reducing and Clustering Hierarchy Algorithm |
| 3D | : Three Dimensional |
| LOG | : Laplacian of Gaussian |
| LBP | : Local Binary Patterns |
| PIL | : Python Imaging Language |
| API | : application programming interface |
| RMSProP | : Root Mean Square Propagation |
| RAM | : Random Access Memory |

# 1. INTRODUCTION

Deep learning is becoming increasingly popular across the world, and many applications based on it are being developed. It is used in image processing, natural language processing, large data analysis, internet search, and other areas.

Practically all deep learning approaches include a deeply layered neural network. Deep learning methods are typically divided into two categories: deep belief networks (DBNs) (Le Roux & Bengio, 2008), and convolutional neural networks (CNNs) (Yuda et al., 2020). The deep belief network (DBN) is a basic, layered network with probabilistic neurons as hidden units. The convolutional neural network (CNN) is a multilayer network with a convolution layer between neural network layers. The neural network output is processed by the max pooling operation in the convolution layer (O'Shea & Nash, 2015). To properly train the deeply layered network, the network is initialized using the statically organized methodology or the lightweight learning method in both methods (Acharya et al., 2017). As I said before, these methods are nearly well-established and have been applied to commercial applications. And the other neural network architectures must be used to make more progress. Several computer vision and pattern recognition systems emphasize training a hierarchy of features for visualization (Danuser, 2011).

The vast majority of deep convolutional neural network (CNN) designs currently in use are designed to gain supervision features, while only a few methods for learning without supervision have evolved. To comprehend the fundamental framework of data, traditional unsupervised learning of features algorithms (Reis & Housley, 2022) nearly exclusively concentrate on using the accessibility of unlabeled data from training images. The Self-Organizing Map (SOM) learning method (Giraudel & Lek, 2001; Kohonen et al., 1996) stands out as one of the many remarkable learning methods that utilize neighborhood functions to master the structure of the high-dimensional data space.

In contrast to supervised CNN structures, unsupervised deep convolutional neural networks receive less attention. Similar to Gabor filters (Joni-Kristian Kämäräinen, 2003), orientation-sensitive filters are learned through sparse coding and unsupervised learning from natural images (Nguyen-Phuoc et al., 2019). A variety of unsupervised deep learning methods, such as K-means (Trevino, 2016), PCANet (Chan et al., 2015), ScatNet (Feng et al., 2021), and SOMNet (Hankins et al., 2018), are available to design a simple deep network.

Developing algorithms that can solve problems computationally is one of the primary objectives of the discipline of computer engineering. Such algorithms have been developed in a major way by human intelligence. However, there are problems whose solutions appear to exceed the capabilities of human intelligence. Additionally, some problems are simple for people to resolve but difficult to express formally. These problems are frequently referred to together with the name artificial intelligence ("Artificial Intelligence Abstracts," 1987).

Differential convolutional neural networks are the fundamental building block of deep learning systems is the process of convolution, which is carried out by swiping several filters over the input image. It offers the ability to extract visual patterns from the supplied image. As a result, the more feature maps the structure produces, the more characteristics the classifier gathers (Sarıgül et al., 2019).

Through a different deviation computation, differential convolution maps are utilized to examine the directional patterns within pixels and the areas surrounding them. It's important to note that in mathematical differentiation, the sequence change is taken into account by figuring out how different the pixel activations are from one another (Qu et al., 2020). A technique to explore the Diff-CNN is to merge differential calculus theories with convolutional neural networks (CNNs) (Sarıgül et al., 2019). It refers to a neural network that incorporates differential calculus in its convolutional layers or during the training process (Sarıgül et al., 2019). This approach can create a neural network that adjusts its parameters in real time according to the input data or shifts in the data distribution.

The differential convolutional neural network aims to transfer feature maps containing directional activation differences to the next layer. This technique takes the idea of how convolved features change on the feature map into consideration (Abd El Kader et al., 2021; Sarıgül et al., 2019). In a sense, this process adapts the mathematical differentiation operation into the convolutional process. This property increases the classification performance without changing the number of filters.

Also, the benefit of differential convolution is the ability to extract more features without adding more convolutional layers by raising the depth of a single convolutional layer. An additional convolution is performed using a differential convolutional layer without the use of any additional trainable parameters (Lei et al., 2018; Sarıgül et al., 2019).

When compared to conventional CNNs, differential CNNs use pre-defined hyperparameters and differential operators to produce feature maps utilizing normal convolutional feature maps (Qu et al., 2020; Lei et al., 2018).

By performing further modifications to the quantity math calculations, differential convolution assesses the pattern direction of each pixel and its neighbors. Computing the difference between pixel activations allows for an evaluation of successive changes (Sarıgül et al., 2019).

Transfer learning is a machine learning technique where a model developed for a particular task is reused as the starting point for a model on a second, related task. This approach leverages the knowledge learned from a pre-trained model, typically trained on a large dataset, and fine-tunes it for a smaller, domain-specific dataset, thereby reducing the need for extensive computational resources and large amounts of labeled data. It is widely used in fields such as natural language processing and computer vision, enabling faster training and often yielding higher accuracy for specialized tasks (Pan & Yang, 2010).

A teacher-student network is a framework in machine learning where a large, complex model (teacher) guides the training of a smaller, simpler model (student). The teacher model, typically pre-trained, provides soft labels or logits that encode richer information about the data distribution compared to hard labels. The student learns to mimic the teacher's predictions, achieving comparable performance with reduced computational resources and model size. This technique, also known as knowledge distillation, is widely used to compress deep neural networks for applications in resource-constrained environments (Hinton, Vinyals, & Dean, 2015).

In this PhD thesis, a novel type of neural network hybridization approach based on the relationships among sequential generations is introduced. The proposed hybridization topology is called intergenerational interaction neural networks (IINNs) .The hypothesis behind this method is that “the presence of a guiding father model enables the son model to succeed quicker and better than the others”. This philosophy can be extended by including incorporating additional ancestors, such as grandfathers and great-grandfathers. Unlike the teacher-student model, the proposed approach involves pre-trained father or ancestor models working together with the son model. In both in training and application phases of the son neural network, father or ancestor parts do not update their weight values.  The proposed approach was implemented by using a Self-Organizing Map (SOM) as the father and a Differential Convolutional Neural Network (DiffCNN) as the son. In this configuration, the SOM serves as the guiding father, independently pre-trained before connecting the son. The proposed NN by using the trained SOM during each training step of the model, the encoded inputs from the SOM were used, and the flattened output was concatenated with the encoded images before being passed through the fully connected layers. This adaptation reduced the complexity of the convolutional layers while improving convergence speed and overall performance. The results demonstrate that the SOMdiffCNN outperforms existing methods, achieving state-of-the-art accuracy on all datasets. By introducing this work, the aim is to bridge the gap between traditional hierarchical learning methods and a more collaborative, intergenerational approach, advancing both the conceptual framework and practical applications of deep learning. And all the implementations were executed in PYTHON environments.

## 1.1. Research Description

This thesis introduces a novel neural network hybridization approach called Intergenerational Interaction Neural Networks (IINNs), inspired by the concept of intergenerational knowledge transfer. The model leverages that “the presence of a guiding father model enables the son model to succeed quicker and better than the others”. This philosophy can be extended by including incorporating additional ancestors, such as grandfathers and great-grandfathers. Unlike the teacher-student model, the proposed approach involves pre-trained father or ancestor models working together with the son model. In both in training and application phases without updating its weight values.

The hybrid architecture, termed SOMDiffCNN, integrates a pre-trained Self-Organizing Map (SOM) as the father network and a Differential Convolutional Neural Network (DiffCNN) as the son network. The SOM, trained independently, provides encoded outputs that are concatenated with image features and fed into the DiffCNN. This structure reduces convolutional layer complexity, leading to faster convergence and improved performance.

The thesis evaluates the SOMDiff-CNN model across six different datasets datasets: MNIST, Fashion-MNIST, Birds, STL10, CIFAR-10, and CIFAR-100. Results demonstrate superior classification accuracy, faster convergence, and reduced training complexity compared to existing methods such as Deep Convolutional Self-Organizing Maps (DCSOM), Differential CNNs (DiffCNN), and standard CNNs. Key achievements include significant performance improvements within the first 10 training epochs, achieving up to 84% faster convergence and state-of-the-art accuracy across all datasets.

This research advances the field of deep learning by proposing a novel intergenerational learning framework, bridging traditional hierarchical learning models and collaborative multi-generational approaches. The developed methodology holds significant potential for applications in real-time image recognition tasks, such as autonomous driving, medical imaging, and the automotive industry.

## 1.2. The Research Goal

The primary aim of this PhD research is to propose and evaluate a new hybrid neural network the Intergenerational Interaction Neural Network SOMdiffCNN architecture, to enhance the performance, convergence speed, and computational efficiency of deep learning models. The presence of a guiding father model enables the son model to succeed quicker and better than the others achieving high accuracy on multiple image datasets. The research also aims to demonstrate the effectiveness of this approach by comparing the proposed model's performance with existing state-of-the-art methods, such as DCSOM, DiffCNN, and CNN, across various benchmark datasets. Also, this research aims to bridge the gap between traditional hierarchical learning methods and a more collaborative, intergenerational approach, advancing both the conceptual framework and practical applications of deep learning. Through this work, the research seeks to advance the field of deep learning by introducing a new, efficient approach to neural network training and model deployment.

## 1.3. Research Outline

The thesis follows a general methodology illustrated in Figure 1.1 and comprises six chapters:

**Chapter 1:** This chapter presents a summary of the research, including an introduction to the Study and its objectives. It also outlines the structure and key components of the Thesis.

**Chapter 2:** This chapter presents a comprehensive overview of relevant literature, including Academic journals, conference proceedings, and thesis.

**Chapter 3:** This chapter provides an overview of the research material and methodology, Including datasets, data preprocessing, supervised and unsupervised learning, classification Analysis, statistical analysis methods Such as Euclidean and Manhattan distance, ensemble learning, Teacher-Student Network, deep learning Techniques such as SOM, CNN, DCAE, DiffCNN, DCSOM, and SOMdiffCNN. We also discuss the Implementation and tuning steps for these Methods, as well as software Environment implementation and utilization of Hardware and frameworks, The Structured Models, and the Evaluation Metrics used in the Thesis.

**Chapter 4:** In this chapter, we will discuss and conclude the results of the proposed method, SOMdiffCNN. We will evaluate these methods on six image datasets and compare it with the results of other deep learning methods, namely DCSOM, CNN, and DiffCNN. The purpose of this comparison is to determine the effectiveness of the accuracy performance of these methods.

**Chapter 5:** This chapter provides a brief overview of the key findings, conclusions, and recommendations of the study.

**Chapter 6:** This chapter contains the references and appendices used in this research

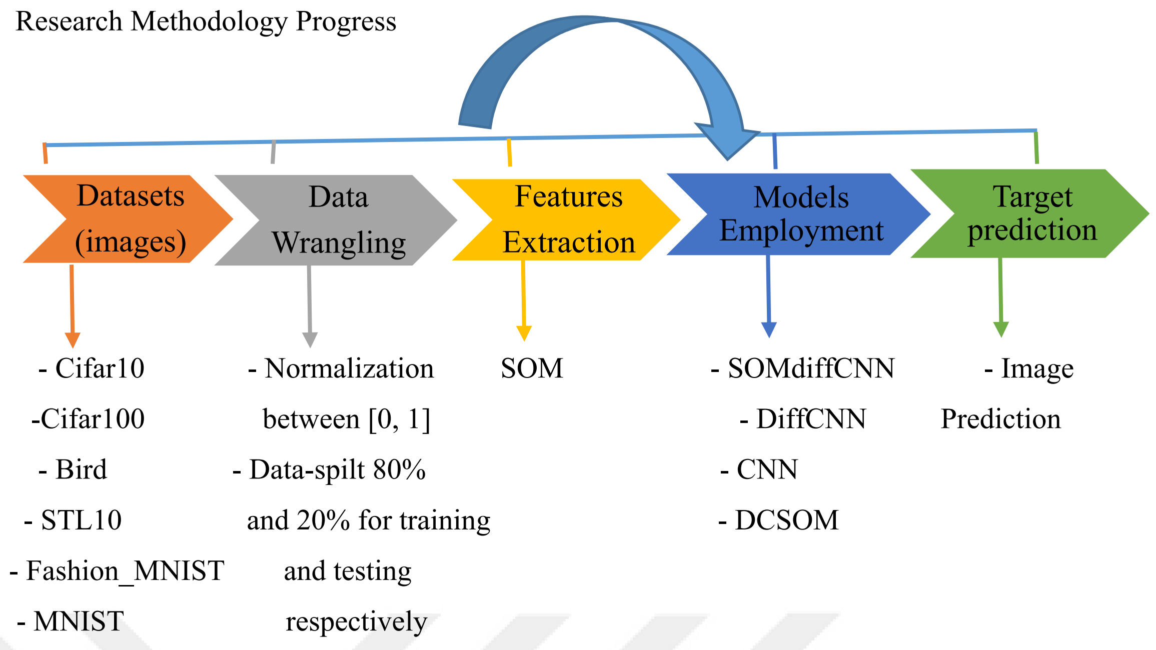

Figure 1.1 Demonstration of general research methodology progress(Source: created by the author)

# 2. PRELIMINARY WORK

Several studies have explored the concept of the hybrid networks for optimizing neural network training and deployment. One significant contribution by Hinton et al. (2015) introduced the concept of knowledge distillation, where the soft outputs from the teacher network are used as additional training signals for the student. This approach demonstrated how smaller networks could achieve comparable performance to larger ones while requiring less memory and computational power. Another study by Romero et al. (2014) extended this idea by introducing FitNets, where intermediate representations from the teacher network were also used to guide the student. This method allowed deeper but thinner student networks to be trained effectively, highlighting the adaptability and scalability of the teacher-student framework.

Another notable work in the teacher-student paradigm was conducted by Yim et al. (2017), who introduced a method called flow-based knowledge distillation. Instead of using only output logits, this approach focused on transferring the flow of activation maps between layers of the teacher and student networks. By capturing and mimicking the relationships between these layers, the student was able to gain a more comprehensive understanding of the learning process. Furthermore, Zagoruyko and Komodakis (2016) explored attention-based transfer, where the attention mechanisms of the teacher model were distilled to the student, further enhancing performance in tasks such as image classification and object detection.

Recent developments in transfer learning have been driven by advancements in pre-trained deep learning architectures. The introduction of models such as ResNet (He et al., 2016) and BERT (Devlin et al., 2019) has transformed the landscape by enabling the transfer of powerful, general-purpose feature representations to a wide range of tasks. These models, trained on large datasets like ImageNet or massive corpora of text, significantly reduce the time and resources required for training on new tasks.

In computer vision, models such as EfficientNet (Tan & Le, 2019) have refined transfer learning by balancing performance and computational efficiency. Techniques like fine-tuning and feature extraction are commonly used to adapt these pre-trained models to specific tasks. In NLP, transfer learning has seen groundbreaking progress with transformer-based models like GPT (Radford et al., 2018) and its successors. These models leverage unsupervised pretraining on extensive text data followed by supervised fine-tuning on task-specific datasets, achieving state-of-the-art results across a wide array of applications.

A seminal contribution to transfer learning was made by Bengio et al. (2012), who highlighted the power of unsupervised pretraining in deep learning models. This foundational work demonstrated how features learned from large-scale datasets could be fine-tuned to address domain-specific tasks with fewer resources. Yosinski et al. (2014) further explored this concept by analyzing the transferability of features across layers in convolutional neural networks (CNNs).

Their findings revealed that lower-layer features are more general and transferable across tasks, while higher-layer features are task-specific and require fine-tuning for optimal performance.

Another key advancement in transfer learning was the introduction of domain adaptation techniques (Pan & Yang, 2010). These methods aimed to align feature distributions between the source and target domains, enabling effective knowledge transfer even when the two domains differ significantly. This concept laid the foundation for many modern transfer learning approaches used in applications such as image classification, natural language processing (NLP), and speech recognition.

There are a variety of traditional approaches that may be used to map or project high-dimensional datasets into a two-dimensional space to see the distribution of huge data sets. Such a technique is known as multidimensional scaling (MDS) (Torgerson, 1952; Leeuw & Heise, 1982), and its frequently used variation is called Sammon's projection (Sammon, 1969). These traditional techniques and their mappings take a long time since they require a lot of processing for huge data sets.

Hiba Mzough et al. developed an efficient 3D CNN model using differential operators in the differential deep-CNN architecture for glioma brain tumor classification using T1-Gado magnetic resonance sequences. The model merges local and global contextual information with reduced weights. The model uses intensity normalization and adaptive contrast enhancement to control heterogeneity and uses data augmentation for successful training. The model outperforms other models, achieving an overall accuracy of 96.49% using the validation dataset (Abd El Kader et al., 2021).

A study by researchers at Hebei University of Technology and the University of Pittsburgh has developed a differential convolutional neural network (differential-CNN) to automatically identify six fetal brain standard planes (FBSPs) from non-standard planes. The study aims to overcome difficulties in detecting fetal brain tissue features due to the non-mature nature of fetal brain tissue and limited labeled image data due to high collection costs. The differential-CNN framework uses differential operators to derive additional feature maps from the original CNN without increasing the number of convolution layers and parameters. The results showed an accuracy of 92.93% in identifying FBSPs from 30,000 2D ultrasound images from 155 fetal subjects aged 16 to 34 weeks. The study demonstrates that differential-CNN can be used for automated identification of FBSPs (Qu et al., 2020).

Sarıgül et al. introduce a new convolution technique called Differential Convolution and an updated error back-propagation algorithm, which considers directional changes among a pixel and its neighbors. It improves standard convolution by extracting pattern orientation. The technique was tested on four different experiment sets, showing improved accuracy in the first set the differential convolution technique increased the accuracy value up to 55.29%. The technique outperformed traditional convolution and other compared convolution techniques, demonstrating its efficiency and adaptability to different convolutional structures. This suggests that differential convolution can be used to improve popular deep learning models (Sarıgül et al., 2019).

Wang et al .presents a hybrid differential evolution (DE) algorithm called DECNN, which aims to automatically evolve the structures and parameters of deep convolutional neural networks (CNNs) for image classification. The DECNN method refines an existing effective encoding scheme for variable-length CNN architectures, develops new mutation and crossover operators for variable-length DE to optimize hyper parameters, and introduces a second crossover to evolve the depth of CNN architectures. The method is tested on six widely-used benchmark datasets and compared to 12 state-of-the-art methods. The DECNN method outperforms IPPSO in terms of accuracy, demonstrating the potential of DE for improving efficiency in designing deep CNNs. The authors propose refining the existing effective encoding scheme used by IPPSO to break the constraint of predefining the maximum depth of CNNs, designing and developing new mutation and crossover operators for the DECNN method, and integrating a second crossover operator to produce children representing CNN architectures with different lengths. The DECNN method is competitive to state-of-the-art algorithms and outperforms IPPSO in terms of accuracy (Wang et al., 2020).

A study investigated the anti-cancer effects of nutritional components on lung cancer cell lines A549 and HLC-1. Results showed that iron treatment inhibited A549 cell growth, while ferroptosis may correlate with cancer cell death. Asiaticoside, a natural compound, showed anti-inflammatory, antioxidant, neuroprotective, and wound-healing properties. A CNN model was used to differentiate between sarcoidosis and lymphoma. A study on asthmatics found age and annual concentration of PAHs as independent factors for the annual decline of FEV1. Understanding the impact of air pollutants on lung function and asthma is crucial ("P3-1: Differential Model of the Deep Convolutional Neural Network between Sarcoidosis and Lymphoma in 18F-FDG-PET/CT," 2021).

Openshaw & Turton discusses the development of Kohonen net-based methods for large spatial dataset classification on a 256 processor Cray T3D parallel supercomputer. The paper aims to reduce multivariate complexity in census data by utilizing the power of the seventh fastest supercomputer. The Kohonen self-organizing map type of unsupervised neural network offers the best levels of performance, as demonstrated by an empirical evaluation of various classifiers. The paper demonstrates the potential of this new type of spatial classification technology for improving the classification of 1991 Census data for Britain (Openshaw & Turton, 1996).

Kaski et al. created a new SOM-based methodology. This approach, known as WEBSOM, is designed specifically for investigating large document collections. Textual documents are organized on a map, which contains capabilities that are helpful for searching across large document collections. The main goals of this study are to decrease the order of magnitude of an application for processing massive document collections, discover a new technique for creating statistical models of documents, discover some "shortcuts" for the SOM algorithm, such as enhancements to the quick creation of large document maps, adopt a different map-initialization technique, and implement a different winner-searching technique (Kaski et al., 1998).

Jaakko Hollmén et al. presents a general framework for learning Self-Organizing Maps, which store probabilistic models in map units. The authors show that minimizing the Kullback-Leibler distance between the unknown true generator of data and empirical models minimizes the negative log probability of the data with the empirical model. The framework is applied to learning multiple user profiles from calling data from a mobile communications network and can be used for fraud detection (Jaakko Hollmén et al., 1999).

Authors Lawrence et al. describe a hybrid neural network solution for face recognition that combines local image sampling, a self-organizing map neural network, and a convolutional neural network. This technique can quickly classify data and just needs quick normalization and preprocessing. It consistently performs better in classification than the eigenfaces technique. Without taking into account invariance to high degrees of rotation or scaling, the research focuses on recognition with variable facial detail, expression, and stance. The authors investigate geometrical feature-based face identification techniques using the ORL database, which contains a collection of faces taken between 1992 and 1994 (Lawrence et al., 1997).

The Self-Organizing Map (SuSi) Python package was introduced by Riese et al., 2019, offering supervised regression and classification for hyperspectral data. SOMs, a weakly represented artificial neural network, are used in unsupervised learning but rarely in supervised learning. The SuSi framework is the first Python package to provide both unsupervised and supervised SOM algorithms for easy usage. It can perform on small and large datasets without significant overfitting. The training process includes initialization, random input data points, best-matching units, learning rate, neighborhood function, and weight matrix modification (Riese et al., 2019).

Li Yuan from the University of Rhode Island has submitted a Master's Theses on implementing Self-Organizing Maps (SOMs) with Python. The project aims to provide Python users with the advantages of the POPSOM package, which can perform functionality beyond model construction and visualization. The study includes migrating the POPSOM package from R to Python, refactoring the source code, and improving the package by adding normalization options. FORTRAN is also embedded to accelerate model construction. The study was a collaborative effort with several committee members, including Dr. Lutz Hamel, Dr. Natallia Katenka, Dr. Austin Humphries, Dr. Orlando Merino, Lorraine Berube, and the author's family (Digitalcommons@uri & Yuan, 2018.).

Adam Coates and Andrew Y. Ng from Stanford University have studied K-means clustering as a fast alternative training method for learning deep hierarchies of features from unlabeled data, particularly images. The authors discuss recent results and technical tricks needed for effective use of K-means clustering for large-scale representations of images. They connect K-means to other well-known algorithms to clarify when K-means can be most useful and convey intuitions about its behavior for debugging and engineering new systems. K-means is useful for learning features due to its speed and scalability, but sparse coding is a better performer in many applications (Coates & Ng, 2012).

Zhao & Zhang introduces a new background modeling method called the stacked multilayer self-organizing map background model (SMSOM-BM), which enhances representative ability and automatic parameter learning for complex scenarios. The method uses deep learning to extend the existing single-layer self-organizing map background model, resulting in a strong representative ability and automatic determination for most network parameters. It also introduces an over-layer filtering process for efficient layer-by-layer training. Implemented using the NVIDIA CUDA platform, the method shows superior performance in real-time performance. Background modeling is crucial for motion detection tasks and modern video surveillance applications (Zhao & Zhang, 2015).

Aly & Aly introduces a deep learning framework for recognizing gestures in Arabic sign language, regardless of the signer this framework employs hand semantic segmentation, hand shape feature representation, and deep recurrent neural networks. For semantic segmentation, DeepLabv3+ is utilized, while CSOM is used for hand-shape features. The Bi-directional Long Short-Term Memory (BiLSTM) recurrent neural network is responsible for recognizing the sequence of extracted feature vectors. This framework surpasses current methods for a signer-independent testing strategy, effectively addressing challenges in hand segmentation, hand shape feature representation, and sequence classification in sign language recognition systems (Aly & Aly, 2020).

Convolutional neural networks (CNN) have been developed in a variety of forms to take advantage of different image modification methods using unsupervised learning. The authors Chan et al.(2015), applied the Principle Component Analysis (PCA) algorithm to learn various filter bank levels and, by cascading this operation, developed a straightforward deep neural network architecture (Chan et al., 2015).

The authors (Tan et al., 2005). Propose a local probabilistic approach to face recognition techniques, extending template-based methods. It utilizes the SOM rather than Gaussians to learn the subspace representing each individual.  Two strategies are suggested: one involves training a single SOM map for all samples, while the other involves training a separate SOM map for each class. Additionally, a soft nearest neighbor ensemble method is proposed to identify unlabeled subjects. The proposed method demonstrates highly robust performance against partial occlusions and varying expressions. Face recognition technology can be applied in surveillance, information security, access control, smart cards, and law enforcement. (Tan et al., 2005).

Hankins et al. suggested SOMNet, an unsupervised approach to image classification using SOM-centered filters to train a collection of non-orthogonal filters, through the discretized representation of neurons. This non-orthogonal alternative to PCANet delivers similar performance. But does not suffer from the same constraints. The authors show that a simple adjustment in the binarization process decreases the dimension of the final feature vector, leading to more filters and deeper structures. They also demonstrate a hybrid method that uses generative Markov random fields as filters for clustering, offering further diverse features in a data-driven deep-learning methodology. (Hankins et al., 2018).

Almabdy & Elrefaei's paper examines the performance of a pre-trained CNN combined with a multi-class support vector machine (SVM) classifier and transfer learning, using the AlexNet model for face recognition. The research focuses on CNN architectures that have achieved top results in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), including AlexNet and ResNet-50. The findings reveal that the CNN model outperformed most advanced models in accuracy, achieving an accuracy range of 92% to 94% across various databases, and demonstrating up to a 39% improvement in recognition accuracy. (Almabdy & Elrefaei, 2019).

Aly et al., a revised version of the hypercolumn model (HCM) network is introduced to enhance face recognition accuracy. The HCM network is structured hierarchically, inspired by self-organizing maps and the visual cortex. The proposed method suggests a change in the feature extraction layer, with a variable dimension for each map, and the quantity of neurons in each map is determined through automated analysis of the training data. This improved HCM network addresses three key issues: dimensionality reduction, invariance, and network parameters. The ORL face database was used to demonstrate the efficacy of this model, which outperformed both neurocognition and standard models (Aly et al., 2008).

The author introduces an enhanced version of the deep SOM algorithm, known as Extended-DSOM, which enhances the learning algorithm by making it completely unsupervised and modifying the architecture to capture features at various resolutions in the hidden layers. The E-DSOM offers improved classification accuracy, generalization capability, and reduced training time. It outperforms DSOM in classification accuracy by up to 15% and reduces training time by up to 19% across all datasets. The study focuses on exhausting SOMs as an unsupervised deep-learning methodology for image classification, overcoming limitations in high-level feature abstraction because of their shallow structure (Wickramasinghe et al., 2019).

This paper presents a parallelizable Deep Self-Organizing Maps (PD-SOM) framework for image classification, designed to enhance the computational efficiency of DSOMs while preserving their performance. The PD-SOM framework provides three key benefits: unsupervised learning for image classification, advanced feature abstraction, and reduced computational expense during training, all while achieving high accuracy. It was tested on the MNIST handwritten digit dataset to maximize accuracy using unlabeled data in a computationally efficient way. Despite its computational demands, PD-SOMs demonstrate notable accuracy improvements over conventional SOMs. (Wickramasinghe et al., 2017).

The rapid expansion of autonomous industrial environments has heightened the demand for advanced video surveillance, especially for detecting complex human movements. While current methods rely on supervised deep learning for human activity recognition (HAR), they face challenges with unlabeled video streams. This article introduces a novel adaptation of the Growing Self-Organizing Map (GSOM) to overcome these limitations. The new approach integrates traditional deep learning concepts, incorporates hierarchical and multi-stream learning, and features a transience property within the algorithm. The effectiveness and applicability of this model are validated through tests on three benchmark video datasets (Rashmika Nawaratne et al., 2020).

Aly et al discuss the usage of Gabor-based face representation for face recognition, which requires large amounts of data. The authors propose a nonlinear projection method using self-organizing maps, utilizing the Multiple Self-Organized Gabor Features (MSOGF) process to symbolize the input image. A novel local matching algorithm is introduced for classifying unlabeled data. The proposed system employs multiple SOMs to understand the distribution of GF with a MAX filter applied to handle local distortion. A self-organizing map (SOM) is trained using Gabor features. The Local Similarity Matching (LSM) algorithm compensates for local feature changes using a Gaussian function (Aly et al., 2010).

Aly & Tsuruta introduce a novel face recognition approach utilizing hierarchical self-organized Gabor features (HSOGF) as a feature extractor. The HSOGF method uses Gabor wavelets, which are resilient to changes in illumination, rotation, and scale, to capture nonlinear data and represent it in a compact feature space. The method uses two-layer self-organizing maps (SOM) to construct a feature map from Gabor filters at each position in the image, reducing computational complexity while maintaining recognition accuracy. The paper discusses setting filter parameters in Gabor filters, including orientation, Gaussian radius, aspect ratio, and a biologically motivated MAX filter to reduce feature shifting in the Gabor response image ( Aly & Tsuruta, 2009 ).

Kulak et al. offer a comprehensive perspective on SOMs and Stochastic Neighbor Embedding (SNE). As data visualization techniques. Both algorithms can result from a common scientific context and can be quantitatively compared on different datasets. The study demonstrates that SOMs adjust neuron weights while keeping points fixed in the 2D space, whereas SNE maintains fixed neuron weights and optimizes point positions. SNE serves both as a visualization tool and a data representation method. The study anticipates that integrating these algorithms will pave the way for future research that capitalizes on their respective strengths. (Kulak et al., 2022).

Wang et al. introduce a novel deep supervised quantization approach utilizing SOMs to address the Approximate Nearest Neighbor (ANN) search problem in multimedia and computer vision. The method combines CNN and SOMs into a single deep architecture, aiming to reduce discrepancies between similar and dissimilar image pairs. It extracts deep features and quantizes them into appropriate nodes within the SOM. Experiments on publicly available standard datasets demonstrate the method's superiority over current ANN search methods, and it can be easily adapted for classification and visualization tasks with minimal modifications. (Wang et al., 2019).

The authors Sakkari & Zaied, introduce a new Unsupervised DSOM algorithm for feature extraction, which is akin to existing multi-layer SOM architectures. This algorithm includes a splitting process, alternating self-organization layers, a ReLU rectification function, and an abstraction layer that combines convolution and pooling. The self-organizing layer comprises multiple maps, each targeting a local sub-region of the input image. The most active neurons are organized into a second sampling layer to form a new 2D map. ReLU is applied multiple times, altering the size of the splitting window and the displacement step on the reconstructed input image each time. (Sakkari & Zaied, 2020).

Aly & Almotairi suggested a new deep unsupervised network called DCSOM for robust handwritten digit recognition. The network employs a series of convolutional SOM layers trained one after another to capture multiple levels of features. The ND-SOM grid uses a competitive learning algorithm to extract abstract visual features. The topological arrangement of features helps manage local transformations and deformations in visual data. Experiments with the MNIST handwritten digit database demonstrate that DCSOM outperforms current state-of-the-art methods in handling noisy digits and other image variations. (Aly & Almotairi, 2020).

Sakkari et al. present a deep self-organizing map model (Deep-SOMs) designed for automated feature extraction and classification from large-scale data streams. The model is based on abstraction, allowing patterns to be extracted from raw data. It includes three hidden self-organizing layers, along with input and output layers, each with multiple SOMs. The model is distinguished by its unique layer architecture, SOM sampling method, and learning approach. It has been validated on large datasets like the Leukemia dataset and SRBCT, outperforming many existing algorithms for image classification. The model is implemented in a distributed architecture in Map Reduce model and Spark (Sakkari et al., 2017).

Split-brain autoencoders are a modified version of the traditional autoencoder architecture used for unsupervised representation learning. These sub-networks extract features from input signals by predicting data channels from one another. They achieve cutting-edge performance on extensive transfer learning benchmarks, allowing for improved transfer to new tasks. Traditional autoencoder models have not demonstrated strong representations for transfer tasks due to their abstraction mechanisms. This work introduces an architectural modification to the autoencoder paradigm, resulting in two separate, concatenated sub-networks trained as cross-channel encoders (Zhang et al., 2017).

The autoencoder (AE) is a key method for anomaly detection, but its ability to distinguish anomalies through reconstruction errors is limited in unsupervised cases. To address this, three new methods are introduced: cumulative error scoring (CES), percentile loss (PL), and early stopping via knee detection. These methods show significant improvements over conventional AE training on image, remote-sensing, and cybersecurity datasets. Unsupervised anomaly detection is crucial in various domains, and these methods prevent AEs from generalizing anomalies across applications, enhancing their reliability in unsupervised DAD (Merrill & Eskandarian, 2020).

Zhu & Zhang introduce a new classification supervised autoencoder (CSAE) based on predefined evenly distributed class centroids (PEDCC) to learn complex data distributions. The method uses PEDCC of latent variables to train the network, ensuring maximum inter-class distance and minimization of inner-class distance. The authors propose a new loss function that combines the loss function of classification and encoding, decoding, classification, and model generalization performance. The main contributions include the PEDCC, which combines classification and autoencoder, and the wavelets loss function, which improves image quality by combining traditional pattern recognition methods. Future research should focus on improving autoencoder accuracy in incremental learning. (Zhu & Zhang, 2019).

Xuejun Zhang et al. propose a novel DDoS attack detection method that trains detection models in an unsupervised learning manner using preprocessed and unlabeled normal network traffic data. The technique uses the Balanced Iterative Reducing and Clustering Using Hierarchies algorithm (BIRCH) to pre-cluster the normal network traffic data and explores an autoencoder (AE) to build the detection model in an unsupervised manner based on the cluster subsets. Experiments on benchmark network intrusion detection datasets KDDCUP99 and UNSWNB15 were conducted to verify the performance of the method. The results showed that the proposed method achieves better performance in terms of detection accuracy rate and false positive rate compared to state-of-the-art models using supervised learning and unsupervised learning. (Xuejun Zhang et al., 2022).

Chen et al. introduces context autoencoder (CAE), a novel masked image modeling (MIM) method, for unsupervised learning. They randomly divide the image into two sets: visible patches and masked patches, in order to improve the encoding quality. From visible patches to masked patches, they make predictions in the latent semantic representation space. For the purpose of designing an unsupervised feature learning network for hyperspectral classification (Chen et al., 2022).

The article "Unsupervised Feature Learning by Autoencoder and Prototypical Contrastive Learning for Hyperspectral Classification" by Cao, Li, and Zhao discusses the growing popularity of unsupervised learning methods for feature extraction. The authors combine traditional contrastive learning and autoencoder to create an unsupervised feature learning network for hyperspectral classification. The proposed network outperforms other comparison methods, maintaining fast feature extraction speed and reducing computing resource requirements. The authors also note that their method separates feature extraction and contrastive learning, allowing more researchers to conduct research on unsupervised contrastive learning.(Cao et al., 2021).

Author Dolgikh presents a new method for enhancing small datasets using unsupervised machine learning. It addresses challenges like insufficient sampling of characteristic patterns, leading to lower statistical confidence and higher error. The authors propose an ensemble of neural network models of unsupervised generative self-learning, which identifies stable clusters of data points in latent representations of observable data. Techniques of augmentation based on identified latent cluster structure are applied to produce new data points and enhance the dataset. This method can be used with small and extremely small datasets to identify characteristic patterns, augment data, and improve classification accuracy in scenarios with strong deficits of labels (Dolgikh, 2021).

Bosch discusses methods for extracting features for student modeling from educational data, specifically interaction-log data, using deep neural networks and unsupervised training. It covers various types of auto encoder networks, including deep, recurrent, variational, convolutional, and asymmetric networks. The implications of training these networks with educational data are discussed, including peculiarities for interaction-log data not commonly encountered in domains like computer vision and natural language processing. The paper also discusses methods for evaluating the network training process and suggests future work in transfer learning and semi-supervised methods (Bosch, 2023).

Park et al. suggest a symmetric graph convolutional autoencoder for unsupervised graph representation learning. It uses a new decoder that creates a symmetric autoencoder form, allowing for reconstruction of node features based on Laplacian sharpening. The autoencoder incorporates signed graphs to prevent numerical instability caused by Laplacian sharpening. A new cost function is developed to find a latent representation and a latent affinity matrix simultaneously, improving image clustering performance. Experimental results show the model is stable and outperforms existing algorithms. The main contributions include the first completely symmetric graph convolutional autoencoder, a new numerically stable decoder form, and a computationally efficient subspace clustering cost.(Park et al., 2019).

Authors Mei et al. presents an unsupervised spatial-spectral feature learning strategy for hyperspectral images using a 3-Dimensional (3D) convolutional autoencoder (3D-CAE). The 3D-CAE uses 3D operations like convolution, pooling, and batch normalization to explore spatial-spectral structure information for feature extraction. A companion 3D convolutional decoder network reconstructs input patterns, allowing all parameters to be trained without labeled samples. Experimental results show the 3D-CAE is highly effective in extracting spatial-spectral features and outperforms traditional unsupervised and supervised feature extraction algorithms in classification applications.(Mei et al., 2019).

There are many research that integrate SOM with CNN. These models typically aim to enhance CNNs with the unsupervised clustering skills of SOMs or to extract deep representations (e.g., CNN codes) and map them into the SOM neural network.

The Table below shows a Summary of the related work.

Table.2.1. Shows a Summary of the related work.

| NO | Years | The Title | The authors | Datasets they used | The algorithms | The Max Accuracy |
|---|---|---|---|---|---|---|
| 1 | 2021 | Differential Deep Convolutional Neural Network Model for Brain Tumor Classification | Abd El Kader, I., Xu, G., Shuai, Z., Saminu, S., Javaid, I., & Salim Ahmad, | Normal and abnormal MR brain images | SVM,KNN and Back Propagation | 99.25% |
| 2 | 2020 | Standard Plane Identification in Fetal Brain Ultrasound Scans Using a Differential Convolutional Neural Network. | Qu, R., Xu, G., Ding, C., Jia, W., & Sun, M. | 2D ultrasound image | K-means, support-vector machine (SVM) , RCM and Back Propagation | 92.93% |
| 3 | 2019 | Differential convolutional neural network. | M. Sarigül, B.M. Ozyildirim and M. Avci. | CIFAR10 and CIFAR100 datasets | Back Propagation | 98.25% |
| 4 | 2020 | A hybrid differential evolution approach to designing deep convolutional neural networks for image classification. | Wang, B., Sun, Y., Xue, B., & Zhang, M. | M NIST, MBI, MDRBI, MRB and MRD datasets | Geneti c Algorithms (GAs) and CNN. | 71% |
| 5 | 2021 | Differential model of the deep convolutional neural network between sarcoidosis and lymphoma in 18F-FDG-PET/CT | Dinesh Patel | 56 patients Image datasets | CNN | 92.9% |
| 6 | 1996 | Development of A parallelized version of SOM | Openshaw and Turton | Geographi cal datasets. | Self-organizing map(SOM) | - |

Table.2.1. Shows a Summary of the related work. (continued)

| NO | Years | The Title | The authors | Datasets they used | The algorithms | The Max Accuracy |
|---|---|---|---|---|---|---|
| 7 | 1998 | WEBSOM – Self-organizing maps of document collections | Samuel Kaski, Krista Lagus, Teuvo Kohonen, and Timo Honkela. | Large Text database. | Computationally efficient algorithms, and random mapping method. | - |
|  | 1999 | Self-organizing map for the probabilistic model. | Jaakko Hollman , Volker Tresp and Olli Simula, | records from calls made with mobile phones | Self-Organizing Map and unsupervised learning. | 86% |
| 9 | 1997 | Face recognition: A convolutional neural-network approach | S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back | ORL Database | Karhunen-Lo`eve transform multi-layer ,perceptron, Convolutional Networks,and Self-organizing map. | 90% |
| 10 | 2019 | Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data. | Riese, F. M., Keller, S., & Hinz, S. | soil-moisture dataset | Self-organizing map (SOM), random forest (RF). | - |
| 11 | 2018 | Implementation of Self-Organizing Maps with Python | Li Yuan | IRIS Dataset | Self-Organizing Map(SOM) | 98% |
| 12 | 2012 | Learning feature representations with k-means | A. Coates and A. Y. Ng | CIFAR10 And STL-10 datasets | K-Means | 82% |
| 13 | 2015 | Stacked multilayer self-organizing map for background modeling | Z. Zhao, X. Zhang, and Y. Fang | A camera captured image dataset. | Self-Organizing Feature Map and NN. | 74% |
| 14 | 2020 | A novel signer-independent deep learning framework for isolated Arabic sign language gestures recognition | S. Aly and W. Aly | Arabic sign language database | BiLSTM,SOM and RNN | 89.59% |

Table.2.1. Shows a Summary of the related work. (continued)

| NO | Years | The Title | The authors | Datasets they used | The algorithms | The Max Accuracy |
|---|---|---|---|---|---|---|
| 15 | 2015 | A simple deep learning baseline for image classification | T.-H. Chan, K. Jia, S. GAO, J. Lu, Z. Zeng, and Y. Ma | MNIST Yale B, AR,FERE T datasets | PCA ,LDA and CNN | 86.66% |
| 16 | 2005 | Recognizing partially occluded, expression variant faces from single training image per person with SOM and soft k-NN ensemble | X. Tan, S. Chen, Z.-H. Zhou, and F. Zhang | AR image database | Self organizing Map(SOM)and A soft knearest neighbor(soft-KNN) | 99% |
| 17 | 2018 | unsupervised feature learning networks for image classification | R. Hankins, Y. Peng, and H.Yin | MNIST | Self-organizing maps(SOM) PCANet and SVM | 86% |
| 18 | 2019 | Deep Convolutional Neural Network-Based Approaches for Face Recognition | Soad Almabdy, and Lamiaa Elrefaei. | ORL,GTA V face, Georgia Tech face, LFW,FLF W face, YouTube face, and FEI faces | Pre-trained CNN AlexNet with SVM, and Pre-trained CNN ResNet-50 with SVM. | 96.63% |
| 19 | 2008 | Visual feature extraction using variable map-dimension hyper column model | S. Aly, N. Tsuruta, R.-I. Taniguchi, and A. Shimada | ORL face database. | SOM, NN, and variable map dimension hypercolumn model. | 91.8% |
| 20 | 2019 | Deep self-organizing maps for unsupervised image classification | C. S. Wicramasinghe, K. Amarasinghe, and M. Manic | MNIST GSA SP-HAR | SOM and CNN | 87.18% |
| 21 | 2017 | Parallelizable deep self-organizing maps for image classification | C. S. Wickramasinghe, K. Amarasinghe, and M. Manic | MNIST | Self-Organizing Map (SOM | 82.88% |

Table.2.1. Shows a Summary of the related work. (continued)

| NO | Years | The Title | The authors | Datasets they used | The algorithms | The Max Accuracy |
|---|---|---|---|---|---|---|
| 22 | 2010 | Hierarchical two-stream growing self-organizing maps with transience for human activity recognition | R. Nawaratne, D. Alahakoon, D. De Silva, H. Kumara, and X. Yu | KTH, Weizmann and UCF11 human activity datasets | Self-Organizing Maps | 94.3% |
| 23 | 2010 | Robust face recognition using multiple self-organized Gabor features and local similarity matching | Aly, A. Shimada, N. Tsuruta, and R.-I. Taniguchi | FERET database | SOM map, Local Similarity Matching (LSM) And KNN. | 93.1% |
| 24 | 2009 | face recognition using hierarchical self-organized Gabor features | S. K. Aly and R.-I. Taniguchi | ORL face database | hierarchical Self-organizing maps, KNN and SVM. | 80% |
| 25 | 2022 | A unified view on Self-Organizing Maps (SOMs) and Stochastic Neighbor Embedding | Kulak, T., Fillion, A., & Blayo, F. | MNIST and and STL-10 datasets. | Stochastic Neighbor Embedding and SOM. | 98.84% |
| 26 | 2019 | Deep scalable supervised quantization by self-organizing map | M. Wang, W. Zhou, Q. Tian, and H. Li | CIFAR-10 and MNIST. | ANN, Self-Organizing Map, CNN. | 98.90% |
| 27 | 2020 | A Convolutional Deep Self-Organizing Map Feature extraction for machine learning | Mohamed Sakkari 1 & Mourad Zaied1 | MNIST and and STL-10 datasets. | SVMs , CNN and SOM | 81.4% |
| 28 | 2020 | Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition. | Aly, S., & Almotairi, S. | MNIST | Self-organizing Map (SOM), Batch learning algorithm and SVM. | - |
| 29 | 2017 | Deep soms for automated feature extraction and classification from big data streaming | M. Sakkari, R. Ejbali, and M. Zaied | MNIST, Leukemia andSRBCT Data sets. | Self-Organizing Map (SOM) | 91.11% |

Table.2.1. Shows a Summary of the related work. (continued)

| NO | Years | The Title | The authors | Datasets they used | The algorithms | The Max Accuracy |
|---|---|---|---|---|---|---|
| 30 | 2017 | Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction | Richard Zhang , Phillip Isola and Alexei A. Efros | NYU-D and ImageNet dataset | Auto-encoder (AE). | 67.1% |
| 31 | 2020 | Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning. | Nicholas Merrill and Azim Eskandarian | Remote-sensing, and cyber security datasets. | Auto-encoder (AE). | 83.5% |
| 32 | 2019 | A Classification Supervised Auto-Encoder Based on Predefined Evenly-Distributed Class Centroids | Qiuyu Zhu a and Ruixin Zhang a | MNIST and Fashion MNIST image dataset | SGD optimization method and Auto-encoder (AE). | 92,89% |
| 33 | 2022 | Exploring Unsupervised Learning with Clustering and Deep Autoencoder to Detect DDoS Attack | Xuejun Zhang1, Jiyang Gai, Zhili Ma, Jinxiong Zhao, HongzhongMa3, Fucun He and Tao Ju | KDDCUP99 and UNSWNB15datasets | K-means, DBSCAN and Clustering Using Hierarchies algorithm (BIRCH) | 96.7% |
| 34 | 2022 | Context Autoencoder for Self-Supervised Representation Learning | Xiaokang Chen, Mingyu Ding, Xiaodi Wang Ying Xin ,Shentong Mo,Yunhao Wang ,Shumin Han ,Ping Luo Gang Zeng, and Jingdong Wang | ImageNet dataset | Context auto-encoder (CAE) and self-supervised learning. | - |
| 35 | 2021 | Unsupervised Feature Learning by Autoencoder and Prototypical Contrastive Learning for Hyperspectral Classification | Zeyu Cao1, Xiaorun Li1 and Liaoying Zhao2 | Salinas, University of Pavia, and Indian Pines datasets. | prototypical contrastive learning and auto-encoder learning method | 95.0% |

Table.2.1. Shows a Summary of the related work. (continued)

| NO | Years | The Title | The authors | Datasets they used | The algorithms | The Max Accuracy |
|---|---|---|---|---|---|---|
| 36 | 2021 | Analysis and Augmentation of Small Datasets with Unsupervised Machine Learning | Serge Dolgikh | small datasets of Covid-19 | Deep autoencoder , and Ensemble of Unsupervised Models. | 91% |
| 37 | 2023 | Unsupervised Deep Autoencoders for Feature Extraction with Educational Data | Nigel Bosch and Luc Paquette. | educational data(Betty's Brain) datasets. | Autoencoder networks, recurrent neural networks, Asymmetric networks, and CNN. | 67.3% |
| 38 | 2019 | Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning | Jiwoong Park Minsik Lee, Hyung Jin Chang, Kyuewang, Lee Jin and Young Choi. | COIL20, YALE, and MNIST datasets and n Pubmed dataset | Convolutional autoencoder, Kmeans, Spectral, and others. | 82% |
| 39 | 2019 | Unsupervised Spatial–Spectral Feature Learning by 3D Convolutional Autoencoder for Hyperspectral Classification | Shaohui Mei, Jingyu Ji, Yunhao Geng, Zhi Zhang, Xu Li, and Qian Du. | Indian Pines Dataset, Salinas Dataset, andSalinas, University of Pavia | Convolutional neural networks, and convolutional autoencoder. | 95.39% |

Previous studies have provided significant advancements in SOMs, CNNs, and teacher-student learning. However, none have successfully integrated these techniques into a unified framework that balances: Deep hierarchical feature extraction, Computational efficiency and structured knowledge transfer.

The primary goal of this research was to develop new novel methods called intergenerational interaction type of neural networks SOMDiffCNN Hybridization approach considering the relation among sequential generation.

Our research redefines the optimization of neural networks by combining Self-Organizing Maps (SOMs), Differential Convolutional Neural Networks (DiffCNNs), and intergenerational learning into a novel hybrid model called SOMdiffCNN. This approach improves accuracy, convergence speed, and overall efficiency. Specifically, this thesis enhances:

1. Feature extraction through SOM-enhanced CNNs.
2. Computational efficiency using differential convolutions.
3. Knowledge transfer via father-son intergenerational learning.This innovative approach

Addresses key limitations in existing models and paves the way for more scalable and adaptable deep learning architectures by Advancing intergenerational learning by integrating structural knowledge transfer beyond traditional logits-based distillation. Fusing SOMs and DiffCNNs for structured self-organization and deep learning. Using differential convolution for enhanced feature representation. And applying intergenerational learning to optimize training and convergence. Making SOMdiffCNN a robust solution for modern AI applications. Bridges the the gap between traditional hierarchical learning methods and a more collaborative, intergenerational approach, advancing both the conceptual framework and practical applications of deep learning. According to the Results, the proposed method should be taken into consideration because they have historically produced the greatest results in image Classification. As compared to the majority of the state-of-the-art models.

# 3. MATERIAL AND METHOD

This chapter focuses on the structure, implementation, and optimization of statistical and deep learning models. It provides a detailed overview of the architecture of these networks and outlines the methodologies used during their training. The chapter also presents a collection of datasets utilized in this research, which serve as benchmarks for evaluating the effectiveness of the proposed models. In addition, it discusses the methodologies for feature extraction, data cleaning, and visualization using the software framework examined in this study. Information about software frameworks and data preprocessing functions is also included. Finally, the chapter explains evaluation metrics and the methods used to assess and compare the quality of classification results obtained in the experiments.

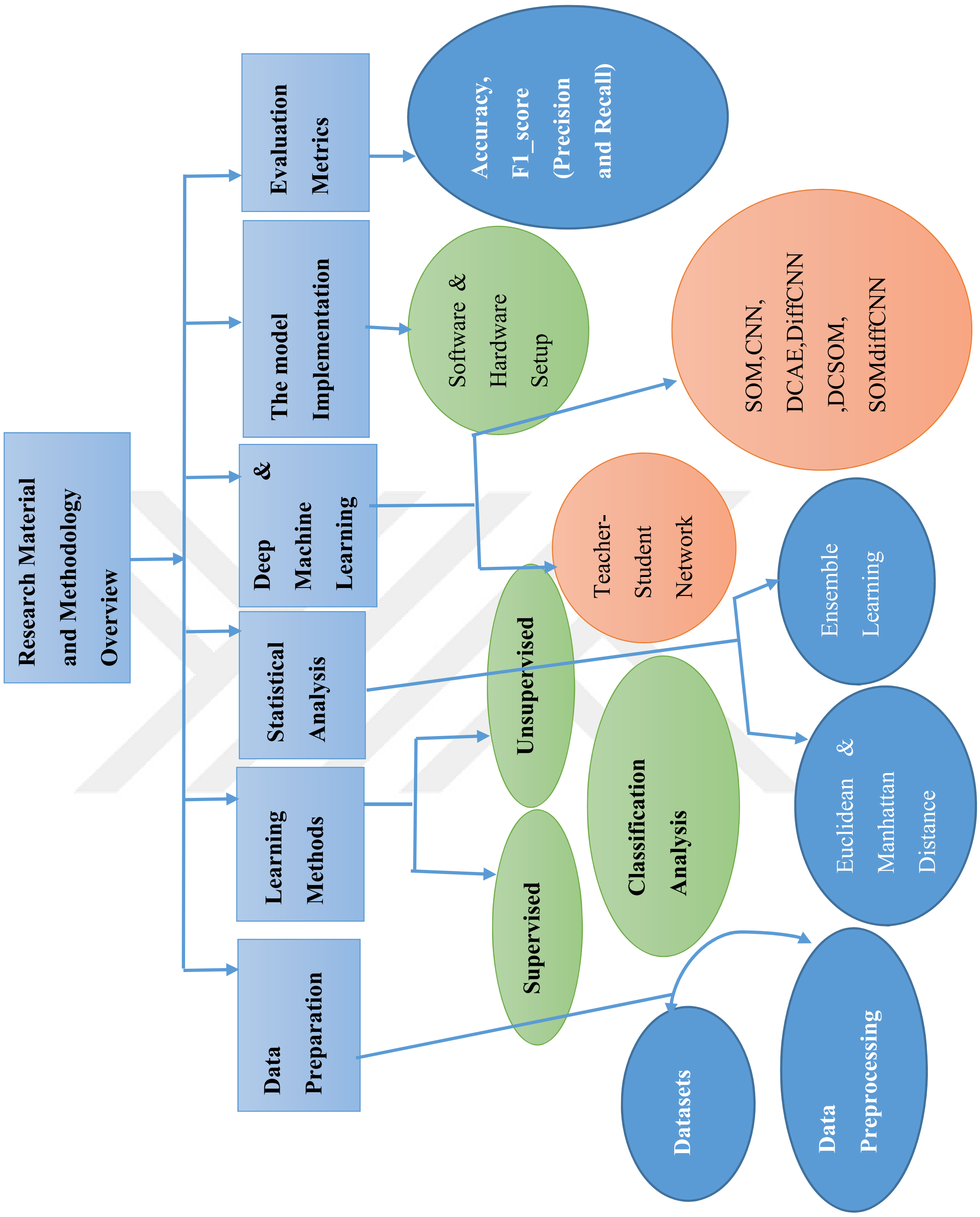

**Figure 3.1** Shows the Research Material and Methodology Overview (Source: Created by the author)

## 3.1 Datasets

The Photographer's Gallery's digital initiative, Data, Set, and Match, is a year-long program that looks for novel ways to share, visualize, and analyze modern image databases. Computer vision is the process of creating an understanding of the information contained in digital images as well as the creation or modification of images using software (C. Rasche, 2022).

In recent years, algorithms have gotten better and better at automatically categorizing photographs, reading license plates, and determining whether tumors are present in medical images the digital photo has evolved into an experiment in new technologies.

The advancement of computer vision has led to methods for improving photos, as various social media sites like Snapchat(Tropp & Baetzgen, 2019) or Facebook Messenger (Smutny & Schreiberova, 2020) now provide a variety of filters. The same methods and developments have also fueled the production of strange psychedelic imagery from deep dreams (Suzuki et al., 2017) or deep fakes ("Detecting and Combating Deep Fakes," 2021), which have now become cultural references. Computer vision algorithms alter how people typically think about what an image is, what it can achieve, and whether or not it can be trusted. They go beyond simple technical advancements. These advancements were made possible by modeling algorithmically how people view, understand, and create images. Computer vision algorithms heavily rely on collections of images called datasets to mimic these cognitive capacities ("Computer Vision: Algorithms and Applications," 2011).

In computer vision, a dataset is a carefully managed collection of digital images that programmers use to test, train, and evaluate the accuracy of their algorithms. It has been suggested that the algorithm picks up new skills from the dataset's samples. Alan Turing outlined what learning meant in this context in 1950. To "point things out" and name them, a dataset in computer vision compiles a series of images that are labeled and used as references for objects in the real world (Lu & Young, 2020).

In this thesis, 6 different image datasets demonstrated the proposed model performance. Including Bird datasets (Mohanty et al., 2020), Fashion _MNIST Datasets (Han et al., 2017), MNIST datasets (Cheng et al., 2020), Cifar10 datasets (Li et al., 2017) Cifar100 (Sun et al, 2017) and STL10 datasets (Ji et al., 2018). All the datasets were normalized to zero mean and unit variance. These studied datasets were gathered from Kaggle.com, and the machine learning repository is publicly available from https://archive.ics.uci.edu/ml//index.phpdatasets,http://cs.joensuu.fi/sipu/datasets/ and https://www.kaggle.com/datasets/dhruvildave their properties are shown in Table 2

### 3.1.1 MNIST dataset

The MNIST dataset comprises 28 by 28-pixel images of handwritten characters ranging from numbers 0 to 9. The dataset consists of 60,000 training images and 10,000 test images, with each row containing 785 values for examples and labels. In this research, a smaller training set of only 3000 photos was used to reduce classifier training time. The accuracy of the algorithms was evaluated using the entire testing set consisting of 10,000 photos. (Cheng et al., 2020).

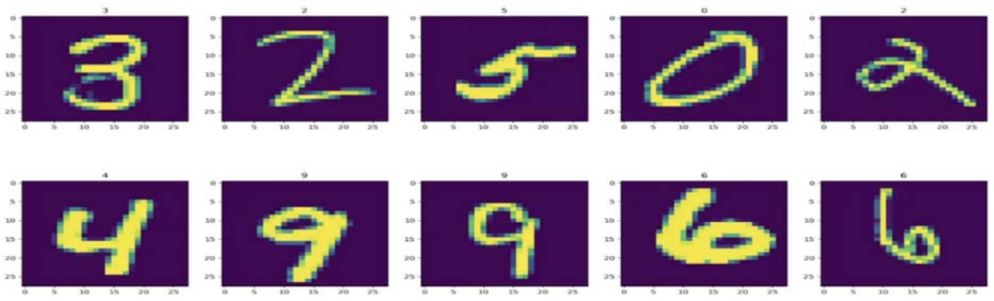

**Figure 3.2.** Shows MNIST datasets. The image is taken from (Krut Patel, 2019)

### 3.1.2 Fashion-MNIST dataset

Fashion-MNIST: A more difficult version of the MNIST dataset, it is a dataset of Zealand’s article images Fashion-MNIST has 60,000 training sets and 10,000 test sets, both of which are in grayscale and sized 28 by 28 pixels. Each image in this research was flattened into a 784-dimensional Vector (Han et al., 2017).

**Figure 3.3.** Shows Fashion-MNIST datasets the image is taken from (Si Lu And Ruisi Li, 2021).

### 3.1.3 Cifer10 datasets

The Canadian Institute for Advanced Research's (CIFAR) CIFAR-10 dataset is a database of images that are widely used to train computer vision and machine learning algorithms. It is one of the datasets that are most frequently utilized in machine learning research. The CIFAR-10 dataset consists of 60,000 32x32 color images organized into 10 groups. The ten groups are cars, deer, dogs, frogs, horses, birds, cats, airplanes, trucks, And ships, every class contains 6000 images (Li et al., 2017).

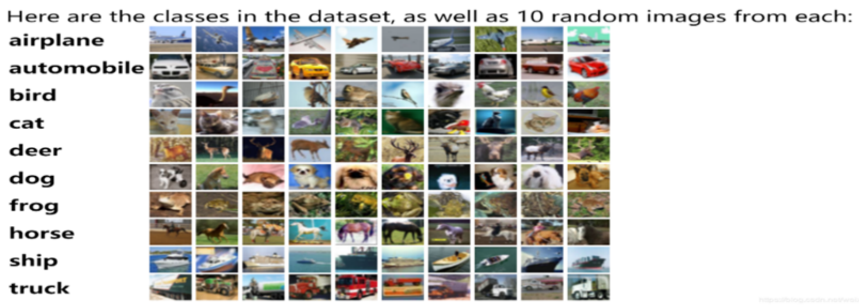
Here are the classes in the dataset, as well as 10 random images from each:
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

**Figure.3.4.** Shows Cifer-10 datasets the image is taken from (CIFAR-10 数据集可视化详细讲解（附代码） | 航行学园, n.d.).

### 3.1.4 Bird dataset

The dataset of bird pictures was gathered in Jordan and contains 525 bird species. The images in the dataset were gathered from scientific sources and certified by the Jordanian Bird Watching Association based on their scientific names (Mohanty et al., 2020).

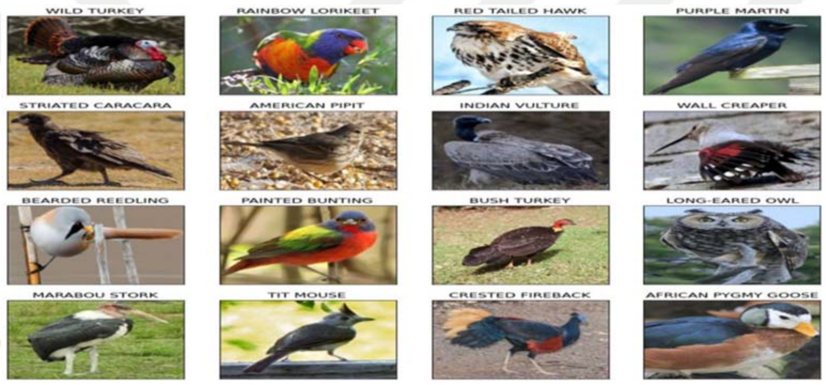

**Figure.3.5** Shows Bird datasets. The image is taken from (BIRDS 525 SPECIES- IMAGE CLASSIFICATION, n.d.).

### 3.1.5 STL-10 dataset

The STL-10 is an image dataset taken from ImageNet that is commonly used to assess unsupervised feature learning or self-taught learning techniques. It includes 13,000 images from 10 item classes (such as birds, cats, and trucks), with 5,000 images partitioned for training and the remaining 8,000 images for testing. All of the images are color with 96×96 pixels in size and it is a very large dataset (Ji et al., 2018).

**Figure .3.6** Shows STL-10 datasets. The image is taken from (STL-10 Dataset, n.d.).

### 3.1.6 Cifer-100 datasets

This dataset is similar to the CIFAR-10 in that it comprises 100 classes with 600 photos. Each class has 500 training images and 100 testing images The CIFAR-100 classifier divides the 100 classes into 20 super-classes. Each picture bears the label "fine" (the class to which it belongs) with a "coarse" label (the superclass to which it belongs) (Bjorn Barz & Joachim Denzler, 2020; Sun et al., 2017).

**Figure 3.7** Shows Cifer-100 datasets. The image is taken from (Aymaz et al., 2022).

## 3.2 Data Preprocessing (Image preprocessing)

Image pre-processing may significantly improve the accuracy of feature extraction and the outcomes of image analysis. The mathematical normalization of data collection, which is a typical step in many feature extraction processes, is an analog to image pre-processing. Descriptive techniques (Krig, 2014).

It is an essential step in computer vision and image analysis tasks. It involves applying various techniques to prepare images for further analysis or processing. Image preprocessing techniques help enhance image quality, remove noise, standardize the format, and extract relevant features (Bieniecki et al., 2007).

Here are some common image preprocessing techniques: Resizing and Scaling: Resizing an image involves adjusting its dimensions to a specific size or aspect ratio ("Image Resizing in Saliency Histogram Domain," 2017). Scaling refers to normalizing pixel values to a specific range (e.g., 0-1) (C. Brian Atkins et al., 2002). Resizing and scaling are often performed to ensure uniformity and reduce computational complexity.

The choice of preprocessing techniques depends on the specific application and the characteristics of the images. It is common to combine multiple preprocessing techniques in a pipeline to achieve the desired image quality and prepare the data for subsequent analysis or machine learning tasks (Krig, 2014).

To ensure that images are comparable in terms of colors, value range, and image size, we must first execute some preprocessing processes before we can extract features from the images.

The preparation processes step in this thesis is taken from openCV and pipelined in the clustimage (Gilewski, 2019). Python provides several libraries for image preprocessing, including OpenCV, scikit-image, and PIL (Python Imaging Library).

These libraries offer a wide range of functions and algorithms for performing various image preprocessing operations.

We applied appropriate image preprocessing techniques, in this research to improve the quality of our image datasets. Which are publically available. To achieve the dataset's greatest score and reduced computation time. When we test it using our different methods a self-organizing map differential convolutional (SOMDiff-CNN), deep convolutional self-organizing map (DCSOM), a differential convolutional neural network (DiffCNN, and Convolutional neural network (CNN).

For the techniques of Rescaling the size of each image in the collection was consistently adjusted to 224×224×155pixels to prevent the impact of image enlargement on the accuracy performance. Additionally, all of the images' input sizes must be the same and correspond with the input dimensions of the proposed model architecture. A similar process was used to combine and reshape two successive sagittal slices to 224×224 pixel sizes. Also, we divided the datasets into two groups, with 25% being used for testing and 75% for training. To balance the OS distribution between the two groups, we used stratified random sampling.

The grayscale is set to either true or false if the images can be properly gray-scaled (as in the case of the Fashion MNST (Han et al., 2017) and MINST (Cheng et al., 2020) datasets). Additionally, the dataset's images were kept in CSV file extension format. With the use of the Euclidean distance metric we used Stochastic Gradient Descent (SGD) (Jonathan &David, 2018), Clasifications are found using the DCSOM, DiffCNN, and standard CNN approaches. With the Accuracy and F_ scores, gets evaluated.

During the preprocessing stages of the images, we used the Python framework to preprocess our six different image datasets image including the OpenCV library (Matthew et al., 2017; Preprocess Images Using OpenCV for Pytesseract OCR, n.d.).

We applied appropriate image preprocessing techniques, in this research to improve the quality of images, extract meaningful features, and enhance the performance of Image Clasifications tasks.

### 3.2.1 The Normalization

As is well known, any kind of issue statement's pre-processing stage includes the normalisation. For data modification, such as scaling down or scaling up the range of data before it is used for future stages, normalization plays a vital function, especially in the areas of soft computing, cloud computing, etc. Min-Max normalization (Ioffe & Szegedy, 2015). Z-score normalisation, and Decimal scaling normalisation (Patro & sahu, 2015) are just a few of the many normalisation methods available. Scaling, mapping, or pre-processing stages are examples of normalisation. From a previously determined range, we can deduce a new range. It can be quite beneficial for purposes of forecast or predicting (Patro & sahu, 2015).

Professionals who deal with a large volume of data are its key users. The data set is altered by the formula, causing the variation to range between 0 and 1. As a result, the largest data unit will be assigned a normalized value of one, and the smallest data unit will get a normalized value of zero in the results of the normalisation calculation. Other data elements will have decimal amounts that range from zero to one (Normalization Formula | Step by Step Guide with Calculation Examples, 2019).

The normalization equation can be expressed mathematically as follows:

$$x \text{ normalized} = \frac{(x - x \text{ mini})}{(x \text{ max} - x \text{ mini})} \qquad (3.1)$$

The following are some significant advantages of the method:

I. Improved accuracy: The machine learning normalization formula aids in raising the accuracy levels of the methods employed in the field as well as in all other kinds of analysis using large amounts of data. It ensures that all factors are equally important and prevents anyone from taking center stage.
II. A better comparison: It makes it easier to compare data from different scales and formats. Every data point falls within the same range, making them similar.

III. Removes duplication: It is especially helpful when a dataset has several dimensions and helps to avoid duplication and inconsistency.

IV. Superior visualization: The approach simplifies the interpretation of the data by making it simple to visualize and display the data in graphics. It becomes simple to create diagrams and graphs.

V. Effective mining of data - The common normalization equation makes algorithmic methods for data mining more precise and successful. The accuracy of data improves when the outliers are reduced. Thus, practical outcomes are obtained (Dheeraj & Ashish, 2023).

Since SOM is sensitive to feature scales, we used the normalization method in this research. Then normalized all datasets to make sure that all features had comparable scales. That is, we take the patch average from all input local patches of size X × X and then divide the result by the standard deviation. There is no need to add the normalization layer to the following SOM Convoluted layer; it is just used on the input image.

## 3.3 **Supervised learning**

Supervised learning is a machine learning approach in which models are trained on labeled data to predict outcomes for new, unseen data. Each training example consists of input-output pairs, allowing the model to learn the relationship between inputs and the desired output. This approach is widely used for tasks such as classification and regression (Bishop, 2006).Key Concepts in Supervised Learning:

1. Data Preparation: Clean and preprocess data, ensuring it is structured with known labels for training. Data is typically split into training and test sets to evaluate model performance effectively (Han, Kamber, & Pei, 2012).
2. Algorithm Selection: Choose from algorithms suited to the task:
   - Linear Regression: For predicting continuous values, like house prices or stock trends (Seber & Lee, 2012).
   - Decision Trees and Random Forests: For interpretability in decision-making tasks (Breiman, 1984).
   - Support Vector Machines (SVM): Effective for high-dimensional data and both linear and non-linear classification (Vapnik, 1995).
   - Neural Networks: Useful for complex patterns, particularly in image and language Processing (Goodfellow, Bengio, & Courville, 2016).
3. Training Process: The model is trained by minimizing the difference between predictions and the actual labeled data using optimization methods, like gradient descent (LeCun, Bottou, Bengio, & Haffner, 1998).

4. Evaluation Assess: the model on unseen data using metrics like accuracy for classification, or mean squared error (MSE) for regression, to understand how well it generalizes (Powers, 2011).
5. Tuning and Improvement: Hyperparameter tuning (e.g., with grid or random search) is used to improve model performance by optimizing the settings for algorithms (Bergstra & Bengio, 2012).

## 3.4 **Unsupervised learning**

Unsupervised learning is a form of machine learning that scans a collection of data for previously undiscovered patterns without the use of labels and with a minimum amount of human supervision (“A Neural Unsupervised Learning Technique,” 1988). Unsupervised learning, often referred to as self-organization, enables the mathematical representation of probability distributions across inputs as opposed to supervised learning, which typically uses a human-labeled dataset (Papageorgiou et al., 2006). It is one of the three primary types of machine learning, along with reinforcement (Horie et al., 2019) and supervised learning (Sotiris Kotsiantis, 2007). A comparable variation called semi-supervised learning employs both supervised and unsupervised methods (Siadati, 2018).

Principal component analysis (Wold et al., 1987) and cluster analysis (Layek & Mukhopadhyay, 1977) are two of the most commonly utilized techniques in unsupervised learning. In unsupervised learning (“A Neural Unsupervised Learning Technique,” 1988), cluster analysis is used to categorize or segment datasets with related properties to determine algorithmic correlations.

### **3.4.1 Process of unsupervised learning**

The following flowchart provides a summary of the general steps we'll take to create an unsupervised learning model (Raina et al., 2009).

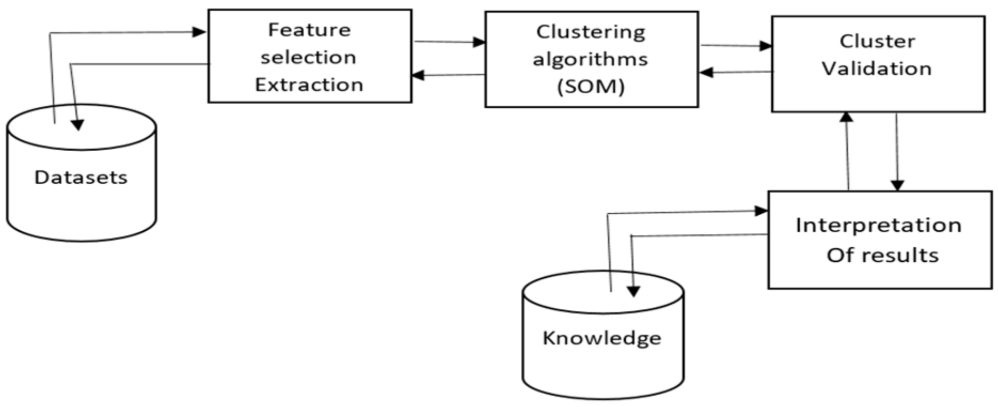
Feature selection Extraction
Clustering algorithms (SOM)
Cluster Validation
Datasets
Interpretation Of results
Knowledge

**Figure 3.9** Shows the chart of the unsupervised learning model(Source: Created by the author).

## 3.5 Classification Analysis

Classification analysis is a fundamental method in machine learning used to assign data points to predefined classes. This technique is applicable across diverse fields such as image recognition, medical diagnosis, spam detection, and more. Classification models are typically trained using labeled datasets, enabling them to generalize and make predictions on new, unseen data (Bishop, 2006). This process typically involves:

Data Preprocessing: Preparing data by cleaning, normalizing, and splitting it into training and test sets to ensure effective model teaching (Han et al., 2012).

Feature Selection/Extraction: Identifying relevant features to improve model accuracy, often using methods like PCA and RFE (Guyon & Elisseeff, 2003).

Model Selection: Choosing the best algorithm, such as Logistic Regression, Decision Trees, SVM, K-Nearest Neighbors, Naïve Bayes, Random Forests, or Neural Networks, based on task requirements (Bishop, 2006).

Model Training: Training the model by minimizing error using optimization techniques, such as gradient descent in neural networks (LeCun et al., 1998).

Evaluation: Assessing the model's accuracy and robustness through metrics like accuracy, precision, recall, F1-score, and confusion matrices, which help gauge performance on imbalanced datasets (Powers, 2011).

Hyperparameter Tuning: Optimizing model parameters using methods like grid search or cross-validation for best results (Bergstra & Bengio, 2012).

Deployment: After satisfactory testing, the model to make predictions on real-world data.

## 3.6 Hardware and Frameworks

The whole computation for this thesis was done on a personal laptop with the following characteristics and a remote machine.

DESKTOP-1MPCSP5: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz, RAM 8, 00 GB, And system type: 64-bit operating system, x64-based processor, Windows 11 Operating System, Pen and touch support with 10 touch points. Utilizing Python programs and a Kaggle framework (Kaggle, 2010). An online portal for data scientists and machine learning enthusiasts is called Kaggle. On Kaggle, users may work together, identify and publish databases, use notebooks with GPU integration, and participate with other researchers to overcome problems associated with data science. In essence, Kaggle Kernels are Jupiter Notebooks that run in a web browser. Consumption of such kernels is unrestricted. Many languages used for programming, including Python and R, among others, are accessible.

Virtual Machine (Google's free cloud service for AI engineers): Google Colab comes with a free GPU.

## 3.7 The Software environment implementation in this thesis

Python belongs to the high-level, examined, multipurpose programming language category. The approach to design places a high priority on code readability and makes extensive use of indentation. Python's benefits for applications founded on machine learning and AI involve its relative simplicity and consistency, the availability of excellent frameworks and libraries for AI as well as deep learning, adaptability, platform autonomy, and a large market (Djokic-Petrovic et al., 2016).

It was produced by Guido van Rossum and released in its initial form in 1991. Its architectural philosophy includes impressive use of extensive emptiness and focuses a heavy emphasis on the accessibility of the coding. Its Object-oriented (OD) programming style and grammar are made to assist programmers in writing clear, understandable codes for projects that are large or small. The enduring popularity of the language as a whole is influenced by these elements (Wikipedia Contributors, 2019).

Flexible typing and collecting waste are features of Python. It allows a range of paradigms for programming, including administrative in nature, object-oriented, and operational programming, as well as organized code (particularly). Python is commonly referred to as a language with "batteries included" because of its large common library. Python was envisioned in the late 1980s as an ABC language substitute. The second version of Python was updated in 2000 with two new features: list comprehension and an explanatory numbering removal scheme. The enduring popularity of the language as a whole is influenced by these elements (About Python, n.d.).

### 3.7.1 PyTorch

PyTorch is a free and open-source framework for the execution of DL models on GPUs and CPUs. It was primarily developed by Facebook's AI Research Lab Group (FAIR) and made available to the public on GitHub in 2017. It provides a platform for research in the fields of computer vision and natural language processing, enabling the creation of novel deep learning architectures or advanced DL architectures with trained models for obtaining cutting-edge results quickly and effectively, regardless of resource limitations. It is renowned for being generally straightforward, easy to use, flexible, easy to debug, and memory-efficient. The SOMdiffCNN architecture was implemented in our thesis using this PyTorch package.

### 3.7.2 TensorFllow

A complete open-source framework developed by Google for building machine learning applications is called TensorFlow. It is a symbolic math toolkit that carries out several operations targeted at deep neural network training and inference using dataflow and identifiable programming (Pulkit Sharma, 2019). It enables the creation of machine learning applications by engineers utilizing a variety of tools, libraries, and neighborhood resources. Google's TensorFlow is perhaps the most popular deep-learning library in the world right now. All of the products offered by Google incorporate machine learning to enhance the search engine, translation, and captioning of images, or recommendations. (Daniel Johnson, 2023). The Google Neuroscience Team created the TensorFlow library as a way to speed up machine learning and deep learning studies. It features many interfaces in different languages like Python, C++, or Java, and it was designed to run on a variety of CPUs, GPUs, and even smartphone operating systems. The first stability version debuted in 2017, but it was first made available in late 2015. Under the Apache Open Source License, it is open source. It is available for usage, modification, and redistribution at a price without any payment to Google (Abadi, 2016).

TensorFlow receives inputs as an array with multiple dimensions called a tensor, allowing you to develop data flow topologies and architectures to specify how data goes through a graph. It enables you to create a diagram of the operations that can be carried out on these inputs, where one end is where the input goes and the other end is where the result comes from(Pulkit Sharmam,2019;Rampasek & Goldenberg, 2016).

There are three components to the Tensorflow building design: handling the data previously, constructing the model, and developing and estimating the model.

Tensorflow is so named because it accepts input in the form of multi-dimensional arrays, or tensors. You can create a "graph" (also known as a flowchart) of the actions that you intend to carry out on that input. The input enters at one end, passes through this network of many processes, and finally exits at another end as the output (Designing TensorFlow Modeling Code for TFX, n.d.).

The Tensorflow library uses a variety of APIs to build deep learning architectures like CNNs and RNNs.TensorFlow, which is based on graph computing, enables the programmer to use Tensorboad to see how a neural network is being built. This tool is useful for program debugging. Finally, Tensorflow is designed for large-scale deployment. The CPU and GPU power it (Google, 2019).

Tensorflow has been imported as TF by using import numpy as np and import Tensorflow as tf. Python is used in our research. The Tensorflow library uses a variety of APIs to build deep learning architectures like CNNs and RNNs. TensorFlow, which is based on graph computing, enables the programmer to use Tensorboad to see how a neural network is being built. This tool is useful for programmed debugging. Finally, Tensorflow is designed for large-scale deployment. The CPU and GPU power it (Google, 2019).

Tensorflow is simple to implement on a large scale, which is why we used it in our research. It is designed to operate on mobile platforms like iOS and Android as well as in the cloud (Johnson, 2023).

### 3.7.3 Keras

Keras is a high-level, easy-to-use API used in developing and educating neural networks. It is a freely available library created in Python that utilizes TensorFlow (Chollet, 2015). It was designed to facilitate quick testing and iteration, and it reduces the entrance hurdle for using deep learning (Chollet, 2015; Pulkit Sharmam, 2019).

TensorFlow needs to be installed before you can use Keras. You may see a minor variation in appearance depending on the operating system you're running, but you can generally use pip, Python's package manager. # first, upgrade pip, such as "pip install --upgrade pip." # then install TensorFlow with "pip install Tensorflow". Then to utilize Keras, just execute the next import statement at the starting point of your script or notebook: "from Tensorflow import Keras (Pulkit Sharma, 2019)."

Because it is the simplest approach to utilizing Keras to generate neural network models, we chose The Sequential API in this study. A neural network can be created by stacking many layer types, one following the other, employing the Sequential class. The Sequential API of Kera supports a wide variety of layer types. Dense layers, fully connected layers, convolutional layers, recurrent layers, and embedding layers are some of the more popular types of layers, while there are many others as well. The layers might be merged to produce effective designs for neural networks. Each layer has the goal of carrying out a certain type of processing on the inputs (Thomas, 2023).

Additionally, we supply a loss function and an optimizer with Kera's when constructing our models.

**There are a variety of possibilities available including Loss functions (Zhao et al., 2017):**

1. Binary Crossentropy: which calculates the cross-entropy between the anticipated and actual binary payments, is a loss function for problems with binary classification.
2. Mean Squared Error (MSE): The sum of the average squared variance among the predicted and actual values is measured by the mean squared error (MSE), a typical loss function for regression issues.
3. Categorical Crossentropy: which quantifies the cross-entropy that exists among anticipated and actual categorical distributions, is a loss function for problems with multiple classes of classification (Zhao et al., 2017).

**The Optimizers:**

- Stochastic Gradient Descent (SGD) is a straightforward optimization approach that computes the gradient of the loss function concerning the parameters before updating the parameters.
- Adam is a learning rate-adapting optimization technique that uses historical gradient data.
- RMSprop is an optimization approach that normalizes gradient updates by using a moving average of squared gradients (Thomas, 2023).

These are only a few of the numerous loss functions and optimizers that Keras offers. The exact issue you're attempting to resolve and the features of the information you have will determine the loss function and optimizer you use. For instance, Mean Squared Error was employed as the loss function as well as Stochastic Gradient Descent (SGD) & Adam as the optimizers in our research to compile our models.

## 3.8 Ensemble Learning

Ensemble learning is a machine learning technique that involves combining multiple models (learners) to create a stronger, more accurate, and robust predictive model. Instead of relying on the performance of a single model, ensemble methods leverage the wisdom of the crowd, tapping into the collective knowledge of multiple models to make more informed and accurate predictions (pinsky, 2018).

Multiple separate models are combined using ensemble learning to improve generalization performance. Deep learning topologies are now outperforming shallow or conventional algorithms in terms of performance. Deep ensemble learning models combine the benefits of the two types of deep learning methods and ensemble learning, improving the generalization performance of the resulting model (Yin et al., 2017).

The fundamental idea behind ensemble learning is that by combining several weak learners, the overall performance can be significantly improved, often outperforming any of the individual models used in the ensemble. A weak learner is a model that performs better than random guessing but may not be highly accurate on its own (Zhou et al., 2002). There are several popular ensemble learning techniques, including Bagging (Bootstrap Aggregating), Boosting (Karel, 2021), Random Forest (Witten et al., 2011), Stacking, and others.

Ensemble learning can provide several benefits, such as increased accuracy, improved generalization, and better resilience to Overfitting. However, it may also be computationally more expensive and require more resources than using a single model (Brownlee, 2021).

When building an ensemble, it is essential to ensure that the base models are diverse, meaning they make different types of errors so that the ensemble can capture a wide range of patterns and improve overall performance. This is often achieved by using different algorithms, varying hyper parameters, or training on different subsets of data (Zhou et al., 2002).

## 3.9 Teacher-Student Network

The teacher-student network concept is based on the idea of using a pre-trained teacher model to assist in training a smaller student model. The goal is to take advantage of the teacher model's performance and knowledge, which includes the subtle patterns or representations learned from the data. This assistance helps the student model achieve similar performance while using fewer parameters and reducing computational complexity (Hinton, Vinyals, & Dean, 2015).

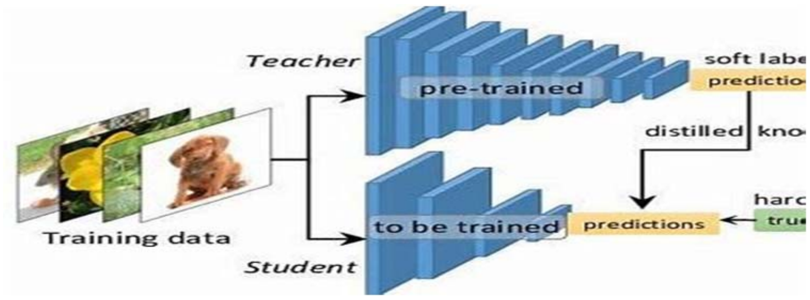

**Figure 3.10**. Shows Teacher-Student Network Structure(ShivamRajsharma.,2021).

### 3.9.1 Essential Elements

1. **Teacher Model**:

A teacher model is usually large, powerful, and complex, such as a deep neural network with many layers. It has already been trained on extensive data and provides high-quality predictions. This teacher model can be state-of-the-art, watch it is computationally expensive but delivers strong performance (Hinton et al., 2015).

2. **Student Model**:

A smaller, simpler, and more efficient model (for example, one with fewer layers or parameters) is often referred to as a student model. This student model is typically trained to replicate the output of a larger, more complex teacher model, enabling it to inherit the knowledge the teacher has gained. While the student may not reach the same performance level as the teacher, it can achieve a comparable level of accuracy while being more computationally efficient (Hinton et al., 2015).

### 3.9.2 Knowledge Distillation

The process of distillation involves training a student model to mimic the output of a teacher model. Instead of relying on the original labels in the training data, the student model learns from the soft targets, which are the output probabilities generated by the teacher model (Hinton et al., 2015).

### 3.9.3 The general steps for knowledge distillation are as follows:

4. Train a large and complex teacher model on your dataset. This model should achieve high accuracy and demonstrate a strong understanding of the problem (Hinton et al., 2015).
5. After the teacher model has been trained, it can be used to make predictions on the training data. The output is usually a probability distribution over the different classes, often referred to as "soft targets." These soft targets provide more detailed information than just the hard labels (Hinton et al., 2015).

6. Train the Student Model: The student model is trained to replicate the soft targets (predictions) provided by the teacher model. The goal is to minimize the difference between the outputs of the student and the teacher. This is typically achieved by using a loss function, such as Kullback-Leibler (KL) divergence, to compare the output distributions of the teacher and student (Hinton et al., 2015).

**3.10 Self-organizing map**

The objective of developing computationally intelligent systems that "enable or facilitate intelligent action in complex and changing environments" is one of the major topics that are currently visible in all engineering sciences areas. The neurobiological-inspired SOM is one of the computational intelligence techniques that is a very effective tool for data processing (Teuvo Kohonen, 2001).

SOMs have found applications in fields such as image processing, natural language processing, and data mining. They are a powerful tool for exploring and understanding complex data patterns in an unsupervised manner (López et al., 2012).

SOM is an ANN technique that uses unsupervised learning to generate a low-dimensional (typically two-dimensional) representation of the input data space (Vesanto & Alhoniemi, 2000). This discrete representation is commonly known as, a "map". A self-organizing map (SOM) may be applied concurrently for multidimensional scaling, grouping, and projection of the multidimensional data set into a lower-dimensional space, like multidimensional scaling. Unlike other artificial neural network methods, SOM employs a neighborhood function to maintain the topological properties of the input space, distinguishing it from other approaches. Developed in 1982 by Professor Teuvo Kohonen from Finland, the SOM algorithm was initially defined as an artificial neural network and is frequently referred to as a Kohonen map (Teuvo Kohonen, 2001; Kohonen, 1990).

A Self-Organizing Map (SOM) is a neural network with a single layer arranged in an n-dimensional grid. While most applications utilize a two-dimensional rectangular grid, some employ hexagonal grids or grids with one, three, or more dimensions. SOMs generate low-dimensional representations of high-dimensional data, maintaining the relationships between similar data points (Vesanto & Alhoniemi, 2000; T. Kohonen, 2014).

Nodes, or neurons, are elements of a Kohonen map. Each node is represented by a location in the map space and a weight vector that matches the data set's dimensions. Nodes are usually arranged in a grid of rectangles or hexagons, where each grid cell represents a node element (Miljković, 2017). The SOM method produces a final topological order of nodes that represents the input space, effectively mapping from a higher-dimensional input space to a lower-dimensional map space (Kohonen, 2013). Each feature from the input space is mapped to the output space, primarily by identifying the node with the closest (smallest distance metric) weight vector, known as the Best Match Unit (BMU), to the input vector. (Laaksonen et al., 2002).

### 3.10.1 Self-Organization Mechanisms in SOMs

The SOM algorithm can be broken down into four main phases (Haykin, 1999):

1. **Initialization:** There are various techniques to initialize the weight vectors of the nodes, with linear initialization and random initialization being the most frequently used (Teuvo Kohonen, 2001; Haykin, 1999). Randomly generating or random initialization of the node's weights can be done by either selecting a random vector for each node or by choosing random vectors from the input space. Whereas with linear initialization, the two most significant principal component vectors the eigenvectors corresponding to the largest eigenvalues are identified first and then used to form a linear two-dimensional space. (Appiah et al., 2012). As the method starts with a well-ordered map that has previously been tuned, linear initialization indicates a quicker convergence of the algorithm. (Valova et al., 2013).
2. **Competition:** Each node estimates the remoteness metric among the input pattern and its corresponding weight vector for each input vector. The winner, also known as the best matching unit (BMU), is determined as the value with the shortest distance (Principe et al., 2009; Appiah et al., 2012; Gunes Kayacik et al., 2007).
3. **The Collaboration:** The winning neuron locates its nearest neighbors who are willing and prepared to cooperate, and then collaborates with them (TAKANASHI et al., 2007). A neighborhood function(Kolasa et al., 2012) that starts as large as the entire map, with the BMU at its canter, eventually shrinks down to the size of a single node to find the BMU's neighbors(Laaksonen et al., 2002).
4. **Adaptation:** Neurons near the BMU adjust their weight vectors based on their proximity to the BMU and the importance of the input pattern transferred to it. The input pattern values and a coefficient related to the distance from the BMU influence neighboring neurons. The input pattern will often be reflected in their altered weights (Lee & Verleysen, 2002).

#### 3.10.1.1The initializing phase

There are many techniques to initialize the weight vectors of the nodes; however, linear initialization and random initialization are the two that are most frequently used (Kohonen, 2001). Either randomly generating values for each node or choosing random vectors from the input space can be used to randomly initialize the nodes' weights. Whereas with linear initialization, the two most significant principal component vectors (those with the largest eigenvalues) are first identified, and these eigenvectors are then utilized to span a linear two-dimensional space. As the method starts with a well-ordered map that has previously been tuned, the linear initialization suggests a faster convergence of the algorithm (Akinduko & Mirkes, 2012).

### 3.10.1.2 The Competition phase

Each node calculates the distance metric between the input pattern and its corresponding weight vector for each input vector. The winner, or best matching unit (BMU), is then determined to be the value with the shortest distance.

The input sequences from the input area might be declared as (x1, x2, x3… xi) D in a D-dimensional space (i.e., a data environment with D attributes), whereas the weight vectors that connect the input elements i and the node j in the output layer (space)

Just a single neuron will emerge victorious from the competition phase. The node denoted as BMU is the one whose weight value is closest to the input value. When calculating the distance between inputs and node weights, a number of distance measurement metrics are used; however, Euclidian Distance and Manhattan Distance are the most widely used (Ghaseminezhad & Karami, 2011).

The function known as the discriminant, which determines the separation between the input vectors x and the weight vector wj for each neuron j using the Euclidean Distance measurement, takes a particular form (Ekanayake & Ranjith , 1994):

$$\text{Euclidean distance} = \sum_{i=1}^{n} (Xi + Wi)^2 \quad (3.2)$$

W is the node's weight vector, while X is the current input vector.

### 3.10.1.3 The Collaboration phase

The winning neuron identifies its own closest neighbors who are eager and ready to work with it and then collaborates with them. A neighborhood function that starts as large as the entire map, with the BMU at its center, eventually shrinks down to the size of a single node to find the BMU's neighbors.

According to Haykin, there is neurobiological proof of lateral connectivity among a group of stimulated neurons. A neuron's neighbor neurons are stimulated when the neuron fires. Neural connections close by are stimulated more than those that are farther away. This implies that the collection of activated neurons is contained inside a topological neighborhood that surrounds the winning node i (Haykin, 1999).

Suppose the total number of stimulated neurons is contained inside the topological neighborhood defined by Nji, which is centered on the BMU i. The function that most successfully satisfies the constraints of the topological neighborhood of SOM is a Gaussian function if the excited neurons (collaborating) are denoted by j and if dist ij suggests the distance between winning neuron i and excited neuron j i (Haykin, 1999).

$$Nji = exp\left(-\frac{dist ji^2}{2\sigma^2}\right) \tag{3.3}$$

The variable σ is the measurement of the width of the geometrical neighborhood function N j, i within the output region. As illustrated in the diagram

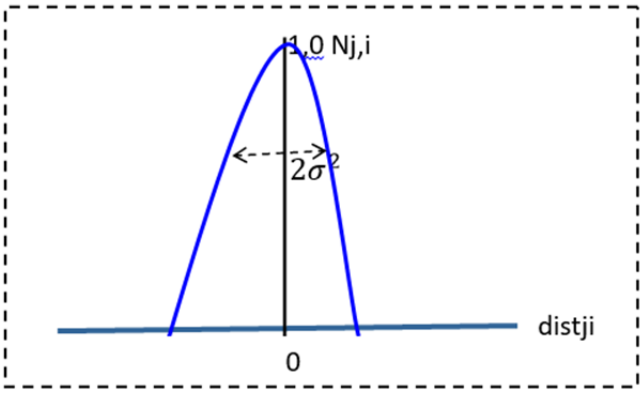

**Figure 3.11** Shows the Gaussian neighborhood function before applying SOM((Source: Created by the author).

The primary and most significant features of this topological neighborhood are as follows:

1. It hits the winning neuron where it has the most value.
2. About the winning neuron, it is symmetric.
3. The distance shrinks to zero monotonically as it approaches infinity.
4. It is translational invariant or independent of where the BMU is located.

In comparison to the rectangle topological neighborhood, this Gaussian topological neighborhood is of higher quality. The introduction of Gaussian topological neighborhood speeds up the convergence of the SOM technique (Yu, & Bavarian, 1993; Obermayer & Schulten, 1992).

The fact that a given topological neighborhood reduces over time is a useful property of SOM. This SOM property can be produced by having the spread factor decrease over time. An exponential decay rate is a frequently used time-dependent function: the following equation is used to calculate the learning rate's decay (Nathan Hubens, 2018):

$$\sigma (t) = \sigma 0 \exp (- \frac{t}{\lambda} ) \qquad (3.4)$$

Where t is the current time step, σ0 is the width of the lattice at time zero, and λ is the time constant. Where t = 0,1,2,3 ...n .The neighborhood shrinks with time after each repetition

$$\Theta (t) = \exp [- \frac{dist^2}{2\sigma^2(t)}] \qquad (3.5)$$

Where $\Theta$ (t) the influence is rate and $\sigma(t)$ is the width of the lattice at time t.and t is the number of iterations (Miljković, 2017; Haykin, 1999).

### 3.10.1.4 The Adaptation phase

Neurons in the area of the BMU adapt by changing their own weight vector to correspond with their nearness to the BMU and the significance of the input pattern that is actually transferred to the BMU. The values of the input pattern and a coefficient proportionate to the distance from the BMU will have an impact on neighboring neurons. The input pattern will often be reflected in their altered weights (Lee & Verleysen, 2002).

SOM requires learning (self-learning), which is essentially a type of adaptive process, in order for the structure of the SOM technique to be self-organized ones.

Output space is made to become self-organized as a result of this adaptation, and as a result, this structure is actually an illustration (mapping) of input space in the output space.

This is accomplished theoretically by modifying the weight vector wj to look like the input vector x.

The neighbors of the winning neuron will also have their weights altered, albeit not to the same extent as the winning neuron. The formula that is applied to update the weight vector is as follows in discrete-time structure:

$$W(t+1) = W(t) + L(t)(V(t) - W(t)) \tag{3.6}$$

Where L is a little variable known as the learning rate that decreases over time, t refers to the time step, and V is the input vector. Once the weights have been updated. For each iteration, the following equation is used to calculate the learning rate's decay.

$$L(t) = L0 \exp\left(- \frac{t}{\lambda}\right) \tag{3.7}$$

Where L is a little variable known as the learning rate that decreases over time. The learning-rate factor is a time-dependent parameter that begins with a starting score of 0 and subsequently declines with passing time t and where t is a further algorithmic time constant. The neighborhood becomes smaller as training continues. At the end of the training, the size of the neighborhoods is zeroed (Haykin, 1999).There are two stages to the adaption process:

1. **Ordering Stages:** The topological ordering of the weight vectors occurs during the ordering process. The SOM method may need to be run up to 1,000 times to achieve it, and the neighborhood and learning rate parameters should be carefully chosen (Haykin, 1999). Learning rate progress L must begin with a value of roughly 0.01. The neighborhood function should first fence in every neuron within the map before gradually reducing this collection of neighbors (Haykin, 1999).
2. **Convergence Stages:** the feature map is adjusted throughout this time, becoming able to accurately and statistically quantify the input space. When the neighborhood function Nj, i is small enough to just contain its immediate neighbors, it can eventually become zero. Learning rate parameters should be carefully chosen; they shouldn't reach zero and should always be about 0.001; if not, they might accumulate to local minima (Haykin, 1999).

### 3.10.2 The SOM Common Topologies

The self-organizing map is made up of a two-dimensional array of neurons. Figure 1.5 illustrates this. This has the same number of dimensions as the input vectors (n-dimensional). The map's topology, or structure, is determined. Through a neighborhood connection, the neurons become connected to their neighbors. Typically, the neurons are connected via linear, rectangular, or hexagonal architecture. The lines connecting the neurons in Figure 1.5 represent the topological relationships.

The most popular SOM topologies are in one or two dimensions, although they can also be in three dimensions (Principe et al., 2009; Haykin, 1998; Anderson, 1995). Rectangular and hexagonal grids are the two most frequently used two-dimensional grids in SOMs. For three-dimensional topologies, shapes can include a toroid or a cylinder. According to (Bondarenko & Katsuk, 2007), 1-dimensional (linear) and 2-dimensional grids are illustrated in Figures. 3.12, with the associated SOMs depicted in Figures 3.13 and 3.14.

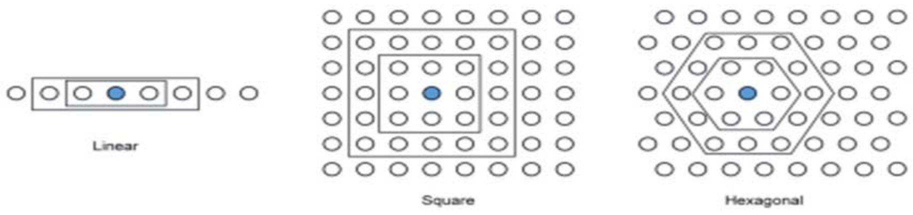

**Figure.3.12**. Shows neuron neighborhoods and the most common SOM grids(Source: Created by the author).

(Image is taken from Brief Review of Self-organizing map/reseachgate.net)

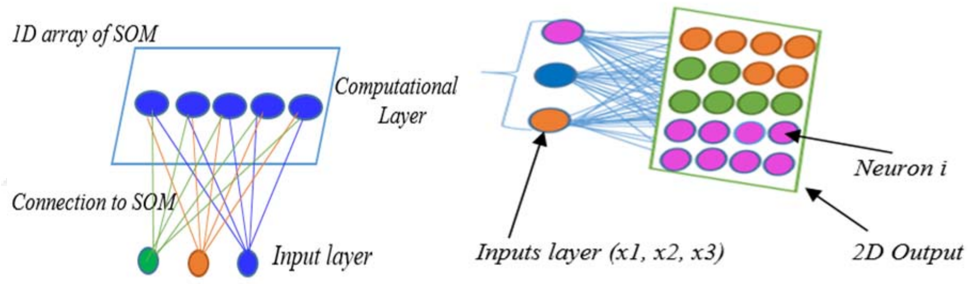

**Figure.3.13**. Shows a 1-D SOM network, **Figure 3.14**. Shows a 2-D SOM network, (Source: Created by the author)

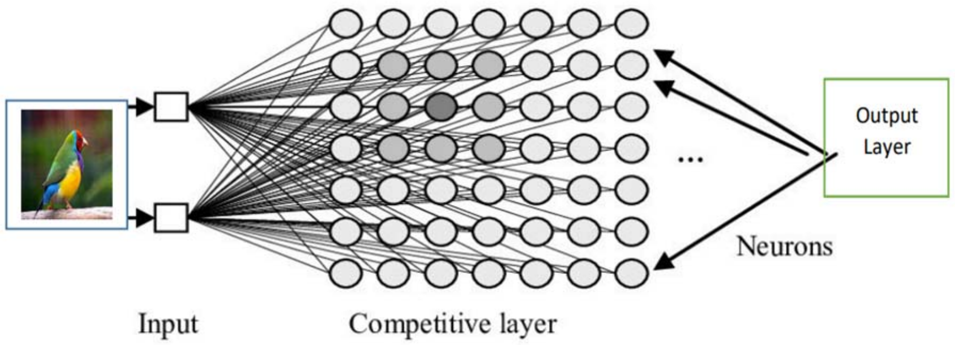

**Figure.3.15** Shows the architecture of the SOM network and the connections between the input and output function regions(Source: Created by the author).

**Summary of the Self-Organizing Map (SOM Algorithm)**

**Algorithm1: SOM Algorithm**

```
Step 1:Ininitialization:
    Randomly initialize the weights $wj$ for each neuron j in the grid.
Step 2:  Sampling:
    Randomly select a vector sample x(n) from the dataset.
Step3:Cocorresponding:
    For each neuron j:
        Calculate the Euclidean distance  as:
            $Distance = \sqrt{\sum_{i=1}^{n} (\mathrm{xi} + \mathrm{wi})^2}$
        Identify the Best Matching Unit (BMU) $i$ as the neuron with the minimum
Step4: Predicting Neighborhood Size:
    Determine the neighborhood size surrounding the BMU
        Calculate the standard deviation $\sigma(t)$ at time $t$ using:
            $h(i,j,t,c) = exp\left(-\frac{\|(\mathrm{i,j}) - c\|^2}{2\sigma(t)^2}\right)$
        Where $(i,j)$ are the coordinates of the neuron and $c$ is the coordinates of the
Step5:  Update Weights:
    For each neuron j in the neighborhood of the BMU:
        Update the weight vector using:
            $wij(t+1) = wij(t) + \alpha(t).hij(t).\left(x(t) - wij(t)\right)$
        where $\alpha(t)$is the learning rate at time t.
Step6: Decay Learning Rate:
    Update the learning rate α(t) using a decay function, typically:
            $\alpha(t) = \alpha 0 \, . \, exp\left(-\frac{x}{\lambda}\right)$
        Where $\alpha 0$ is the initial learning rate and t is the decay time constant.
Step 7:  Looping:
    Repeat steps 2 through 6 until the map ceases to change significantly
End
```

To implement our model, we applied the SOM model for dimensionality reduction and feature extraction from the dataset.

## 3.11 Euclidean Distance

The Euclidean Distance is the "common" distance between two or three points in mathematics that may be calculated by using a ruler. The Pythagorean formula produces the Euclidean distance or Euclidean metric. Euclidean space (or any inner product space) is actually a metric space by utilizing this formula as distance. The Euclidean standard is known as the name of the associated standard (Liberti et al., 2012). If two points in Euclidean n-space are represented in Cartesian coordinates as X=(x1, x2, x3 … xn) and W= (w1, w2, w3 … wn), then the distance from X to W or from W to X is given by:

$$E\_distance = \sqrt{(w1 - x1)^2 + (w2 - x2)^2 + \cdots . (wn - xn)^2} = \sqrt{\sum_{i=1}^{n} (wi - xi)^2} \quad (3.9)$$

## 3.12 Manhattan Distance

The Manhattan distance function calculates the grid-space travel distance between two data points. The total of the differences in the corresponding coordinates of two places is known as the Manhattan distance. Their corresponding coordinates in grid space correspond to the node positions in the grid (Chiu et al., 2016) A point X=(x1, x2, x3 … xn) and a point W= (w1, w2, w3 … wn) are separated by this distance using the following formula:

$$M\_distance = \sqrt{|w1 - x1|^2 + |w2 - x2|^2 + \cdots |wn - xn|^2} = \sqrt{\sum_{i=1}^{n} |wi - xi|} \quad (3.10)$$

The values of the i-th variable at the points x and w, respectively, are xi and wi, where n is the number of variables. The difference between the Manhattan distance and the Euclidean distance is shown in the following figure:

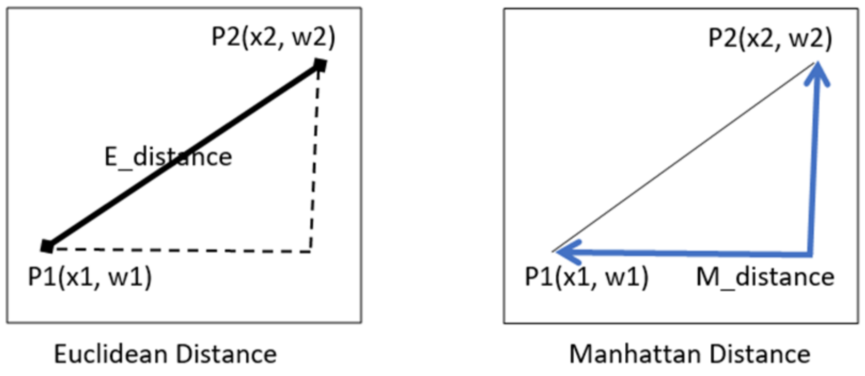

**Figure.3.16**. Shows the difference between Manhattan Distance and Euclidean Distance(Source: Created by the author).

## 3.13 Neighborhood Functions

Most machine learning techniques depend heavily on neighborhood functions, especially those in competitive learning and self-organizing maps (SOM) families. During the learning process, these mechanisms describe the influence or impact that a specific data point (or neuron) in the model has on its nearby data points.

The main goal of Neighborhood functions' is to incorporate the concept of proximity or similarity into the learning process. When given a neighborhood influence, data points in the input space that are close to one another will have a greater influence on one another's learning than data points that are farther apart. This enables the model to self-organize and meaningfully represent the underlying data distribution (Natita et al., 2016).

Typically, the neighborhood function is based on a distance metric that measures how similar two data points are. Euclidean distance (Liberti et al., 2012) and Manhattan distance (Chiu et al., 2016) are two popular distance measures. The neighborhood function is typically stated as a function of distance, with values that decrease as the distance from a particular data point rises. The neighborhood function's shape and size might vary depending on the particular algorithm being utilized (Aoki & Aoyagi, 2007).

In topological spaces, the neighborhood is a fundamental and important mathematical idea. When talking about ANN algorithms, and particularly when referring to SOM, the area surrounding a point (the winning neuron) is a set that fences in certain other neurons nearby. Additionally, the winning neuron in the center can influence the weights of the other neurons in this set, pushing them to move toward the winning neuron's state (Natita et al., 2016).

If X1 is a topological space and P1 is a point in X1 then a neighborhood of P1 is a subset S of X1 that includes an open set U that contains P1 where p1□U□X1 (Kelley & John, 1975).

Different neighborhood functions can be utilized in the case of self-organizing maps, including:

**A. Gaussian neighborhood function**

The Gaussian neighborhood function assigns weights to data points based on their distance from a central point (e.g., a neuron in the model). The influence of data points decreases exponentially as the distance from the central point increases (Natita et al., 2016). The formula for the Gaussian neighborhood function is typically expressed as:

$$h(i,j,t) = \exp\left(-\frac{||\mathbf{x}_i(t) - \mathbf{x}_j(t)||^2}{2\sigma^2(t)}\right) \tag{3.11}$$

Where:

- h (i, j , t) is how data point j affected data point i's learning at time step t.
- The positions of data points i and j in the input space at time step t are represented by the values of x i (t) and x j (t).
- σ(t) is The width parameter at time step t determines how widely the neighborhood spreads. (Natita et al., 2016 ; Yang et al., 2015)

**B. Bubble Neighborhood Function**

The bubble neighborhood function which defines a fixed-size neighborhood surrounding a central point, is more straightforward than the Gaussian function. While data points outside the bubble remain unaffected by learning, those inside the bubble are affected (Natita et al., 2016). The bubble neighborhood function's formula is frequently as follows:

- The influence of data point j on data point i's learning at time step t is represented by the formula h (i, j, t).
- At time step t, the data points i and j are located at xi (t) and xj (t), respectively, in the input space.
- The neighborhood's fixed radius at time step t is represented by radius (t).

But often the Neighborhood range is established using a Gaussian kernel surrounding the winner neuron or a rectangular neighborhood function. The processor must compute the exponential function, making computation of the Gaussian function, as in equation (3.11), significantly more expensive. Rectangular functions can be used to approximate this function with high accuracy while requiring less computation but at the cost of adding more neurons to the update phase of Self-Organizing Maps (Murakoshi & Sato, 2007).

In self-organizing maps, neighborhood functions are essential because they give the network the ability to arrange and represent high-dimensional data in a lower-dimensional space, exposing the underlying structure and patterns in the data. Additionally, they are employed in many clustering methods that rely on the idea of similarity between data points as well as other competitive learning algorithms.

## 3.14 Deep learning

Modern artificial intelligence (AI) systems may now be designed with many billions of basic elements due to significant advancements in computer technology. When these components are correctly established and followed by training, AI may execute tasks that were previously thought to be so extraordinarily difficult that only natural intelligence systems, i.e., human beings, could perform those (LeCun et al., 2015; Benuwa et al., 2016).

Deep learning is primarily to blame for this achievement in AI. While loosely based on biological neural networks, such as those in your brain, artificial neural networks are best thought of as an especially attractive way of specifying a flexible set of functions. They are built out of many basic computational units called neurons. Deep learning uses artificial neural networks as the underlying model for AI (Bengio, Y, 2009; Perspectives and Future Outlook of Deep Learning AI, 2017). In reality, this computation model differs significantly from the one that drives the computer. Deep learning models, in particular, are trained on data from the real world and learn how to solve problems rather than programming a precise set of instructions to solve a problem directly.

Deep neural networks are where deep learning gets its actual power. By organizing a large number of parallel neurons into successive processing layers, the nervous system may develop helpful world representations (Yasaka et al., 2018). Such representation learning is regarded as a defining characteristic of success in artificial and biological intelligence since it turns data into ever-more-refined forms that are useful for executing an underlying goal (Norgeot et al., 2019; Sam, 2013).

Deep learning is a member of a border family of machine learning (ML) techniques based on artificial neural networks with billions of artificial neurons arranged in layers between the input and the output (Amini et al., 2020). The input features that are necessary for discriminating are enhanced by the hidden layers, while unimportant variations are suppressed. Learning can be supervised, semi-supervised, or unsupervised (Good fellow et al., 2016; Bengio, & Hinton, 2015; Schmidhuber, 2015).

Deep learning uses a general-purpose learning approach to find the discriminative characteristics required for detection or classification after being fed a vast amount of unprocessed data. According to (Good fellow et al., 2016; Bengio, and Hinton 2015, and Schmidhuber 2015), the general-purpose learning procedure represents the raw data as a nested hierarchy of features or concepts in many stages between the input and output layers. Each concept is defined in terms of simpler concepts, and more abstract representations are built in terms of less abstract ones. Deep learning can vastly outperform systems that rely on features that are manually created or provided by domain experts by learning the automated discriminative features that are necessary for detection or classification (Valavanis & Kosmopoulos, 2010; Lecun et al., 2015).

Deep learning has produced significant progress in many different areas of artificial intelligence research. Over the past decade. This technology, which developed from previous studies on artificial neural networks, has outperformed other machine learning algorithms in a wide range of complex problems, including those involving image and voice recognition(Das, et al., 2015), natural language processing (Howard & Ruder, 2018; Peters et al., 2018b, 2018a), machine translation, bioinformatics, drug discovery, genomics (Chang et al., 2018; Jiménez et al., 2018), the analysis of unstructured, tabular-type data using entity embedding's(Yury et al.,2022),image classification or clustering (Wouter Van Gansbeke et al., 2020), speech recognition and synthesis (Oord et al., 2016; Xiong et al., 2018), image analysis (Chen, et al., 2020; Farabet et al., 2013), and the reconstruction of brain circuits (Helmstaedter et al., 2013; K. Lee et al., 2019), among others. Recent years have seen the emergence of the first wave of deep-learning applications in pharmaceutical research. These applications go beyond bioactivity predictions and show promise for solving a variety of issues in drug discovery. Additionally, it has influenced our daily lives due to its wide acceptance by major technology companies like IBM, Google, Apple, Microsoft, and Adobe. OpenAI, Deep Mind, and other current deep learning research institutions work to develop secure AI systems that learn how to solve challenges and promote scientific research for everybody (Toshihiro Takahashi, 2018).

Different categories of deep learning approaches have also been developed and implemented by experts in order to solve different kinds of problems (Sutskever, & Hinton, 2012). These categories include convolutional neural networks (CNN or ConvNet),(Zhou et al.,2020), multi-layer perceptron's (MLP) (Zhao et al, 2015), auto-encoders(AE) (Weng et al. ,2016), deep belief networks (DBN) (Hinton, G.,2009), deep Boltzmann machines (DBM) (Salakhutdinov et al,2012), capsule neural networks (Goldani et al., 2021), recurrent neural networks (RNN) (Schuster & Paliwal, 1997), long-short-term memories (LSTM) (Hochreiter & Schmidhuber, 1997), and generative adversarial networks (GAN) (Odena ,2019) and others.

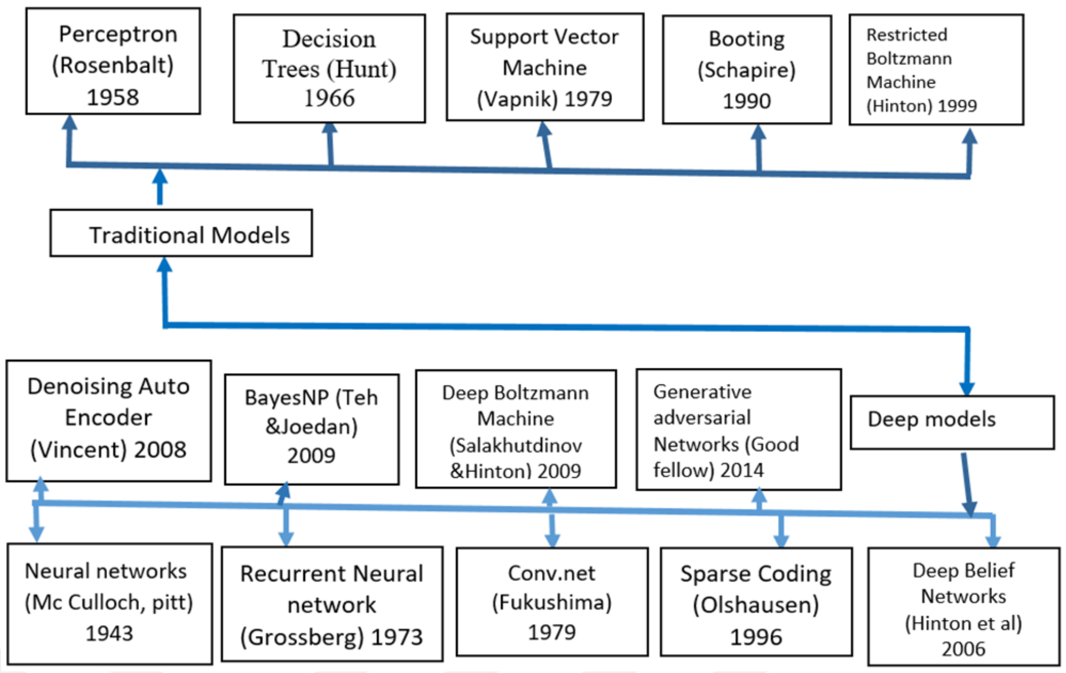
Perceptron (Rosenbalt) 1958
Decision Trees (Hunt) 1966
Support Vector Machine (Vapnik) 1979
Booting (Schapire) 1990
Restricted Boltzmann Machine (Hinton) 1999
Traditional Models
Denoising Auto Encoder (Vincent) 2008
BayesNP (Teh &Joedan) 2009
Deep Boltzmann Machine (Salakhutdinov &Hinton) 2009
Generative adversarial Networks (Good fellow) 2014
Deep models
Neural networks (Mc Culloch, pitt) 1943
Recurrent Neural network (Grossberg) 1973
Conv.net (Fukushima) 1979
Sparse Coding (Olshausen) 1996
Deep Belief Networks (Hinton et al) 2006

**Figure 3.17** Illustrates the progress of Deep Learning Models(Source: Created by the author).

### 3.14.1 Convolutional neural network

Convolutional Neural Networks (CNNs /ConvNet) are a particular kind of artificial neural network (ANN) that have been shown to perform well on a variety of visual tasks (Keiron O'Shea & Nash, 2015; Dorafshan et al., 2018)., such as image classification( Tao & Talab 2017; Copur, Melisozyildirim, & Ibrikci, 2018; Pak & Kim, 2018), image segmentation(Khan et al., 2014), image retrieval(Datta et al.,2008), object detection(EriĢ & Çevik, 2019; Donahue et al., 2014; Sultana et al. ,2020; Ulku & Akagündüz , 2022)., image captioning(Anderson et al., 2018), face recognition(Xie & Lam , 2008), pose estimation(Sun et al.,2019), traffic sign recognition(Miura et al.,2000), speech processing(Hou et al., 2018), neural style transfer, video processing (Ballas et al., 2016; Mathieu et al., 2016) and more.

CNNs are made to deal with data that is presented as several arrays, such as a color picture made up of three 2D arrays that each include the pixel intensities for each of the three color channels (Cheng et al., 2016). With the help of their convolutional filters, they are able to extract information from pictures. The early layers are used to detect edges, the later layers are used to detect sections of things, and the later layers may even be used to detect whole objects, such as faces or other complicated geometrical structures. The convolutional layer, pooling layer, and fully-connected layer are the three primary types of layers that make up CNN and may be categorized according to their capabilities (Frosst, & Hinton, 2017).

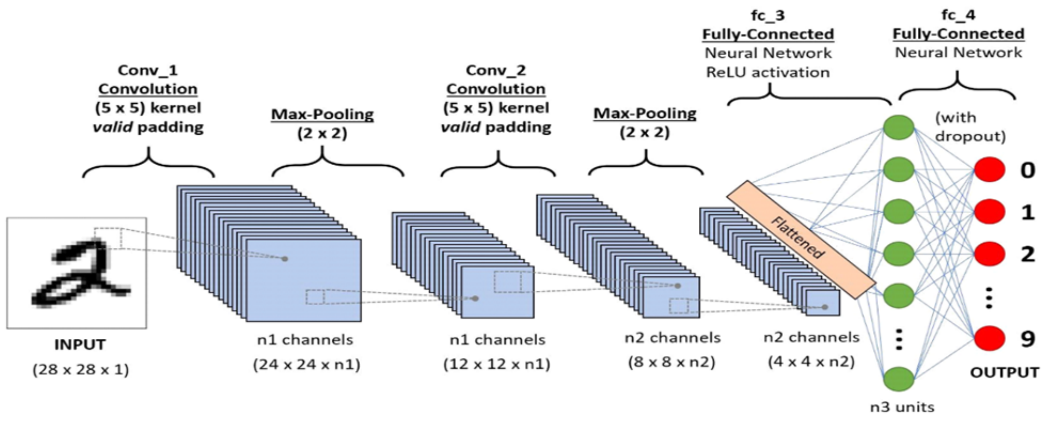

**Figure 3.18** A Convolutional Neural Network Architecture including (convolutional, pooling, and fully-connected layers)(muhammad osman tarig et all.,2020).

### 3.14.1.1 Convolutional layer

A convolutional neural network's basic building block is the convolution process. The parameters of the convolutional layer are a collection of learnable filters (kernels). Every filter has a narrow spatial range (in width and height), but it covers the whole depth of the input volume. Sizes like 3x3, 5x5, and 7x7 are typical filter sizes (Pandian et al., 2022). The total number of input channels is represented by the filter's third dimension. The grayscale image depth is 1, and there are three RGB color channels in the color image (Hamed Habibi Aghdam et al., 2017).

Each filter performs convolution on the input volume across the width and height and computes the dot products between the filter entries and the input at any position during forward propagation. This operation is followed by a nonlinear activation function (sigmoid, tanh, ReLU, etc.) (Ertuğrul, 2018), and the outputs produced are referred to as feature maps. The response of the filter is provided at each spatial point by the feature map, sometimes referred to as an activation map. Displays the filter's reactions at each individual location on the map. There are three erp, padding, stride, and depth that determine output volume (He et al., 2016). The output size is determined by using the formula (Anthimopoulos et al., 2016):

$$(n + 2p - f) / s + 1. \qquad (3.12)$$

Where p is the amount of padding, f is the size of the filters, n is the number of filters, and s is the stride.

### *3.14.1.2 Pooling layer*

The pooling layer technique, also known as subsampling or downsampling, is frequently used by CNNs to decrease the dimension following convolution layers. The filter size and strides are represented by the pooling layer's hyperparameters. The pooling layer with filter size 2 and stride 2 is the most common. Max pooling and average pooling are two common types of pooling layers where the maximum and average values are taken, respectively. More frequently than average pooling, max pooling has been used. The parameters for the pooling layer cannot be learned. The idea behind maximum pooling is that a feature may be discovered because of the high quantity (Yamashita et al., 2018).

#### *3.14.1.3 Fully Connected layer*

The CNN often ends with many fully connected layers after several convolutions and poolings. The output tensor from these layers gets transformed into a vector before more neural network layers are added (Basha et al., 2019). The dropout (Choe & Shim, 2019).a regularization approach can be used in the fully connected layers to prevent overfitting. The architecture's last fully connected layer has exactly the same number of output neurons as the number of classes to be identified (Szegedy et al., 2015).

### 3.14.2 Convolutional Auto encoder neural network

A Convolutional Autoencoder (CAE) is a type of neural network that combines the concepts of convolutional neural networks (CNNs) and autoencoders. It is commonly used for unsupervised learning and dimensionality reduction in computer vision tasks, such as image denoising, compression, and feature extraction (Seyfioglu et al., 2018). The components and working of a Convolutional Autoencoder:

Auto-encoders (AE): is a deep neural network method utilized for unsupervised feature learning with effective data encoding and decoding, typically for data compression and dimensionality reduction (Yousefi-Azar et al., 2017)..

Also, an autoencoder is a type of neural network that aims to encode the input data into a lower-dimensional representation and then decode it back to the original data

The goal of auto-encoders (AE), which are neural networks, aims to duplicate the inputs in the outputs. In order to produce the desired result, they first compress the input into a latent-space representation and then rebuild the output from this representation (Wen et al., 2019). Two components make up this type of network:

Encoder: This part compresses the input data and maps it to a lower-dimensional latent space representation. In a CAE, the encoder typically consists of convolutional layers followed by pooling or downsampling layers, which help to extract relevant features from the input data(Han et al., 2019).

The encoder section of the network compresses the input into a representation of latent space. By using the encoding function h=f(x), it may be expressed.

Decoder: This part takes the compressed representation from the encoder and reconstructs the original input data. In a CAE, the decoder consists of convolutional layers followed by upsampling or De-convolutional layers, which help to reconstruct the data from the compressed representation.

Reconstructing the input from the latent space representation is the goal of the decoder. By using the decoding function r= g (h), it may be expressed (Han et al., 2019).

Combining Convolutional Layers in Autoencoder: In a Convolutional Autoencoder, the convolutional layers in both the encoder and decoder play a crucial role in learning relevant feature representations from the input data. The convolutional filters are trained to capture local patterns in the image, and as the network goes deeper, it can capture more complex and abstract features (Seyfioglu et al., 2018).

#### 3.14.2.1 Working of Convolutional Autoencoder:

1. Data Input: Images or different kinds of data are provided to the CAE as sources. In this scenario, we are using images, which are modeled as 3D tensors (height, width, and channels) (Le, 2015).
2. Encoder: The encoder section of the CAE utilizes a sequence of convolutional and pooling layers on the input data in order to extract pertinent features and minimize its geographic dimensions. An encoder's output is a compressed representation that is frequently referred to as "latent space" or "code." (Han et al., 2019).
3. Decoder: The decoder section of the CAE uses an array of upsampling and convolutional layers in order to recreate the original input data from the compressed version (Han et al., 2019).
4. Loss Function: The CAE is trained using a loss function that measures how comparable the input data and the output of the reconstruction are. On image-based autoencoders, Mean Squared Error (MSE) is a popular option. Which is used in this thesis.
5. Training: In order to decrease the process of reconstruction loss, the CAE trains how to improve the weights of the convolutional filters in the encoder as well as the decoder.

As a preprocessing stage for other neural network architectures or for various downstream tasks, the CAE is very good at learning condensed and understandable representations of the input data (Kamran Ghasedi Dizaji et al., 2017).

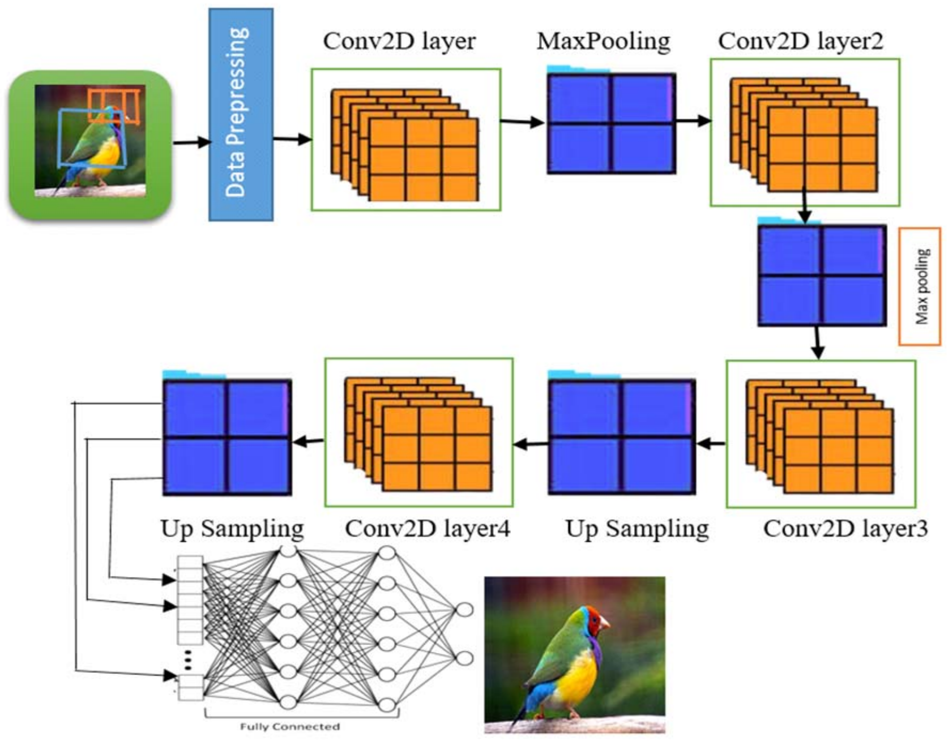

**Figure 3.19** Shows Convolutional Auto-encoders (DCAE) Neural Network Architecture including (convolutional layers for both encoding and decoding) (Source: Created by the author).

The DCAE architecture includes convolutional layers for both encoding and decoding operations to reduce the spatial dimensions and extract features. An encoder with convolutional and max-pooling layers is used. In a decoder, there are layers of upsampling and convolution to reconstruct the original image. The final layer is supposed to be the activation function used to make sure that the pixel values are between 0 and 1. Then, the model is predicted for the image reconstruction with a loss function.

### 3.15 Deep convolutional Self-organizing map

A deep convolutional self-organizing map (DCSOM) is an extension of the self-organizing map (SOM) algorithm that incorporates convolutional layers, enabling it to learn spatial features in a hierarchical manner. It combines the unsupervised learning capabilities of SOMs with the feature learning capabilities of deep convolutional neural networks (CNNs). Is commonly used for tasks of image clustering, visualization, and unsupervised feature learning (Aly & Almotairi, 2020).

The initial design of the deep Convolutional Self-Organizing Map (DCSOM) is done by, the convolutional part of the deep convolutional neural network is tried to be added to the deep self-organizing map. The new hybrid topology has merged the concepts of SOMs and convolution neural networks (CNNs). The hybridization is done by using existing standard learning methods for the two different parts combined together (Sakkari & Zaied, 2020).

### 3.15.1 The Architecture

The architecture of a deep convolutional self-organizing map (DCSOM) typically consists of multiple convolutional layers followed by a SOM layer. The convolutional layers are responsible for learning hierarchical representations of the input data, capturing spatial features at different levels of abstraction. The SOM layer performs the clustering and mapping of the learned features onto a low-dimensional grid (Ferles et al., 2021).

Below is the DCSOM neural network architecture:

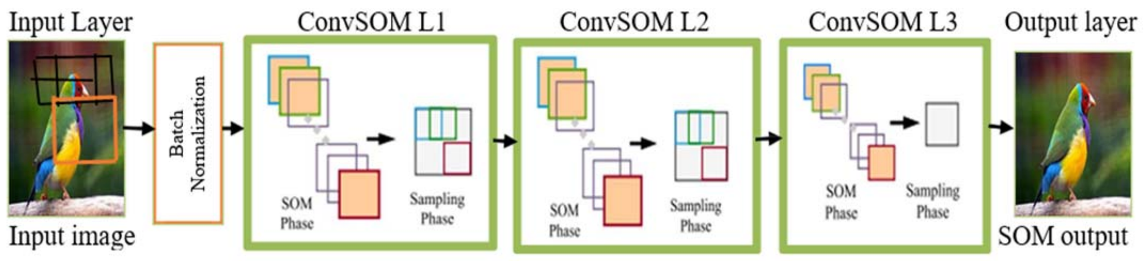

**Figure 3.20** Showing DCSOM neural network architecture Diagram(Source: Created by the author).

### 3.15.2 Convolutional Layers

The convolutional layers in the DCSOM perform local receptive field operations on the input data. They consist of convolutional filters (or kernels) that slide across the input data, extracting spatially local features. Each convolutional layer is followed by a non-linear activation function (e.g., ReLU) and pooling layers (e.g.,Max Pooling) to downsample the feature maps (Aly & Almotairi, 2020).

### 3.15.3 SOM Layer

The SOM layer in DCSOM performs competitive learning and clustering. It consists of a grid of neurons, where each neuron represents a weight vector. The weight vectors are adjusted during training to match the learned features from the convolutional layers. The SOM layer preserves the spatial relationships between the neurons, ensuring that similar features are mapped to nearby neurons (Aly & Almotairi, 2020).

### 3.15.4 Training

The training of a DCSOM involves a two-step process. First, the convolutional layers are trained using unsupervised learning, typically through a pre-training phase using autoencoder learning techniques. Then, the SOM layer is trained using the learned features from the convolutional layers. The training process involves presenting input data, finding the best-matching unit (BMU) in the SOM layer, and updating the weights of the BMU and its neighboring neurons (Aly & Almotairi, 2020).

### 3.15.5 Clustering and Visualization

After training, the DCSOM used for clustering the image datasets is similar to the traditional SOM. Each input data point is assigned to the neuron with the closest weight vector. Additionally, the low-dimensional grid of neurons in the SOM layer is visualized to gain insights into the learned features and their spatial organization (Aly & Almotairi, 2020).

## 3.16 Differential convolutional neural networks

The fundamental building block of deep learning systems is the process of convolution, which is carried out by swiping several filters over the input image. . It offers the ability to extract visual patterns from the supplied image. As a result, the more feature maps the structure produces, the more characteristics the classifier gathers (Sarıgül et al., 2019).

Through a different deviation computation, Differential convolution maps are utilized to analyze directional patterns within pixels and their surrounding areas. It's important to note that in mathematical variation, the change in sequence is taken into account by figuring out how different the pixel activations are from one another (Qu et al., 2020).

A technique to explore the Diff-CNN is to merge differential calculus theories with convolutional neural networks (CNNs) (Sarıgül et al., 2019). It refers to a neural network that incorporates differential calculus in its convolutional layers or during the training process (Sarıgül et al., 2019). This approach can create a neural network that adjusts its parameters in real-time according to the input data or shifts in the data distribution.

The Diff-CNN is designed to pass feature maps with directional activation differences to the next layer. This approach incorporates the concept of how convolved features vary across the feature map (Abd El Kader et al., 2021). Essentially, it adapts mathematical differentiation to the convolutional process, enhancing classification performance without altering the number of filters.

Also, the benefit of differential convolution is the ability to extract more features without adding more convolutional layers are enhanced by raising the depth of a single convolutional layer. An extra convolution is applied using a differential convolutional layer, without adding any extra trainable parameters. (Sarıgül et al., 2019).When compared to conventional CNNs, differential CNNs use pre-defined hyperparameters and differential operators to produce feature maps utilizing normal convolutional feature maps.

By performing further modifications to the quantity math calculations, differential convolution assesses the pattern direction of each pixel and its neighbors. Computing the difference between pixel activations allows for an evaluation of successive changes (Sarıgül et al., 2019; Lei et al., 2018).

Figure 3.16: Displays the predefined feature maps. The difference in one direction is computed for each feature map. As a result, extra feature maps with disparities along various directions are obtained. In contrast to Sarıgül et al., they extend the original approach and add a fixed filter to extract more task-related information (Sarıgül et al., 2019; Qu et al., 2020).

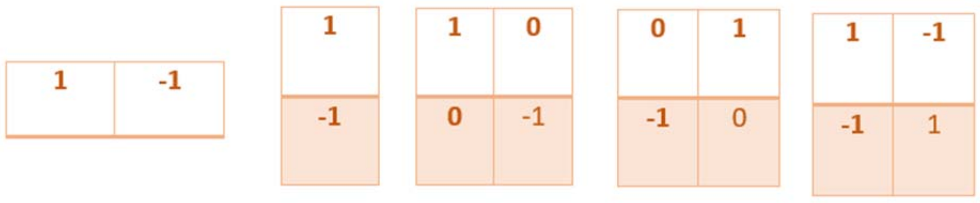

**Figure 3.21** Shows the available predefined filters for Diff-CNN(Source: Created by the author).

The differences in one direction are determined using each of the above filters. As a result of this, supplementary feature maps are produced that incorporate signal differences for each direction.

Utilizing a differential operator, every one of the five feature maps produced from the classic CNN is G2, G3, G4, G5, and G6. Each map's neurons are computed from (3.13) to (3.17) (Qu et al., 2020):

$$G2,i,j = G1,i,j - G1,i+1,j \tag{3.13}$$

$$G3,i,j = G1,i,j - G1,i,j+1 \tag{3.14}$$

$$G4,i,j = G1,i,j - G1,i+1,j+1 \tag{3.15}$$

$$G5,i,j = G1,i+1,j - G1,i,j+1 \tag{3.16}$$

$$G6,i,j = G1,i+1,j+1 - G1,i,j+1 \tag{3.17}$$

if we Assume that (i) and (j) denote the coordinates of the neurons in the convolutional feature maps, G1 has dimensions of (M x N), while G2, G3, G4, G5, and G6 have dimensions of (M - 1)× N, M×(N - 1), ((M - 1)× (N-1),(M - 1)× (N - 1), and ((M - 1) × (N - 1), in that order(Sarıgül et al., 2019; Abd El Kader et al., 2021). Figure 1.5 illustrates the method used to calculate these feature maps.

Once creating the initial feature map through typical convolution, differential equations are employed to produce Diff-CNN feature maps from this original feature map. These differential convolution feature maps can recognize edges and corners, among other things (Qu et al., 2020).

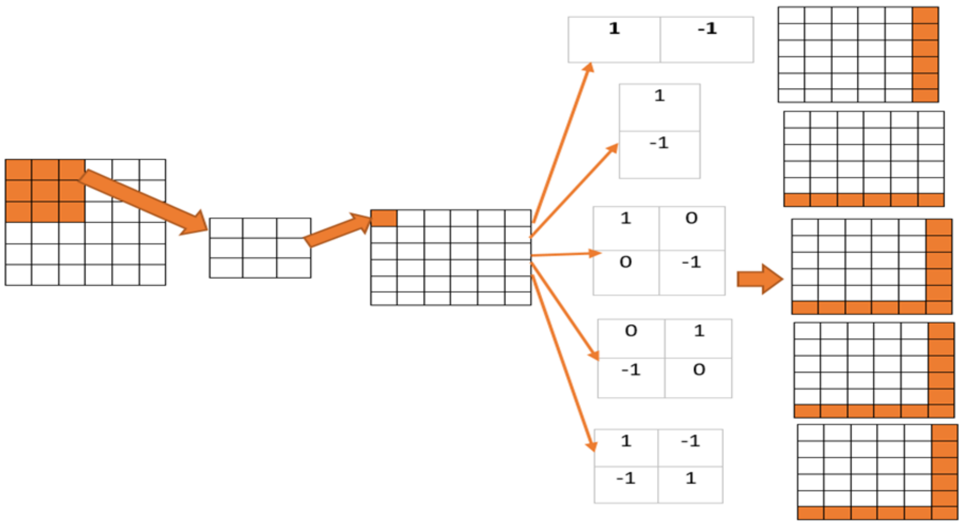

**Figure 3.22** Shows how the differential feature maps are determined(Source: Created by the author).

By utilizing differential feature maps, the Diff-CNN can extract more image features without requiring additional convolution layers. This is evident from the previously mentioned derivation procedure. As a result, the Diff-CNN reduces the complication of the CNN framework and decreases computational demands (Sarıgül et al., 2019).

Back-propagation Procedure: To apply Diff-CNN with the new feature maps, an enhancement to the back-propagation process is necessary. During back-propagation, errors are propagated across each feature map. In the opposite direction. The fixed filter weights specific to each feature map are then multiplied by the errors and added together. The error matrix obtained from the first feature map's related errors is transmitted backward to train the appropriate filter. (Sarıgül et al., 2019; Qu et al., 2020; Lei et al., 2018).

Calculating the derivative of an activation function is useful for finding the gradients of the loss function concerning the network's parameters during the backpropagation process. The gradients act as a guide for the optimization algorithm to update the parameters, resulting in the reduction of loss and enhancement of network performance (Abd El Kader et al., 2021).

We use h1 to represent the error passed to the first map, while h2, h3, h4, and h5 represent the errors transferred to the additional maps. E represents the error matrix. The error computations for the corresponding filter are shown in functions (3.18) to (3.20) (Qu et al., 2020).

$$\begin{aligned} Ei,j \;=\; & h1,i,j \;-\; h2,i,j-1 \;+\; h2,i,j \;-\; h3,i-1,j \\ & + h3,i,j \;-\; h4,i-1,j-1 \;+\; h4,i,j \;-\; h5,i-1,j \;+\; h5,i,j-1 \end{aligned} \tag{3.18}$$

In the given scenario, when i exceeds 1, j is greater than 1, i is less than M, and j is less than N, the neurons situated at the corners as well as the edges do not receive error input from all the adjoining neurons. Equation (3.18) highlights this fact. However, the corner neurons of the map receive an error signal from three of their nearby neighbors, as mentioned in Equation (3.19),(, (Sarıgül et al., 2019).

$$Ei,j = \begin{cases} h1,i,j + h2,i,j + h3,i,j + h4,i,j, & i = 1,j \\ h1,i,j - h2,i,j-1 + h3,i,j + h5,i,j-1, & i = 1,j = m \\ h1,i,j + h2,i,j - h3,i-1,j - h5,i-1,j, & i = n,j = 1 \\ h1,i,j - h2,i,j-1 - h3,i-1,j - h4,i-1,j-1, & i = n,j = m \end{cases} \tag{3.19}$$

The error that will spread to the neurons located at the boundary of the map is estimated in (3.20). Five nearby Neurons deliver error signals to the edge neurons (Sarıgül et al., 2019).

$$Ei,j = \begin{cases} h1,i,j - h2,i,j-1 + h2,i,j + h3,i,j + h4,i,j + h5,i,j-1 \\ \quad i = 1, 1 < j < N \\ h1,i,j - h2,i,j-1 + h2,i,j - h3,i-1,j - h4,i-1,j-1 - h5,i-1,j \\ \quad i = n, 1 < j < N \\ h1,i,j + h2,i,j + h3,i,j - h3,i-1,j + h4,i,j - h5,i-1,j \\ \quad 1 < i < M, j = 1 \\ h1,i,j - h2,i,j-1 + h3,i,j - h3,i-1,j - h4,i-1,j-1 + h5,i,j-1 \\ \quad 1 < i < M, j = n \end{cases} \tag{3.20}$$

## 3.17 Intergenerational Interaction Neural Networks

This method draws inspiration from the conceptual idea that the presence of a guiding father model enables the son model to succeed quicker and better than the others. Embodying the essence of intergenerational interaction within neural network architectures. Unlike the teacher-student model, this approach involves pretrained father or ancestor models work together with the son model. The Self-Organizing Map (SOM) serves as the father, providing pre-trained, high-level feature representations that guide the training of the son. The method was implemented by using a Self-Organizing Map (SOM) as the father and a Differential Convolutional Neural Network (DiffCNN) as the son. In this configuration, the SOM serves as the guiding father, independently pretrained before connecting the son. The proposed approach by using the trained SOM during each training step of the model, the encoded inputs from the SOM were used, and the flattened output was concatenated with the encoded images before being passed through the fully connected layers. This adaptation reduced the complexity of the convolutional layers while improving convergence speed and overall performance

By embedding this intergenerational interaction mechanism, the DiffCNN benefits from reduced convolutional layer complexity and accelerated convergence. The SOM's unsupervised clustering capabilities offer structured guidance, while the DiffCNN's adaptability ensures the model evolves with the data. This synergy mirrors the collaborative dynamic between generations, where foundational wisdom enhances adaptability and innovation. This framework has promising applications in areas such as adaptive learning systems, cross-disciplinary AI models, and the modeling of generational dynamics in real-world scenarios.

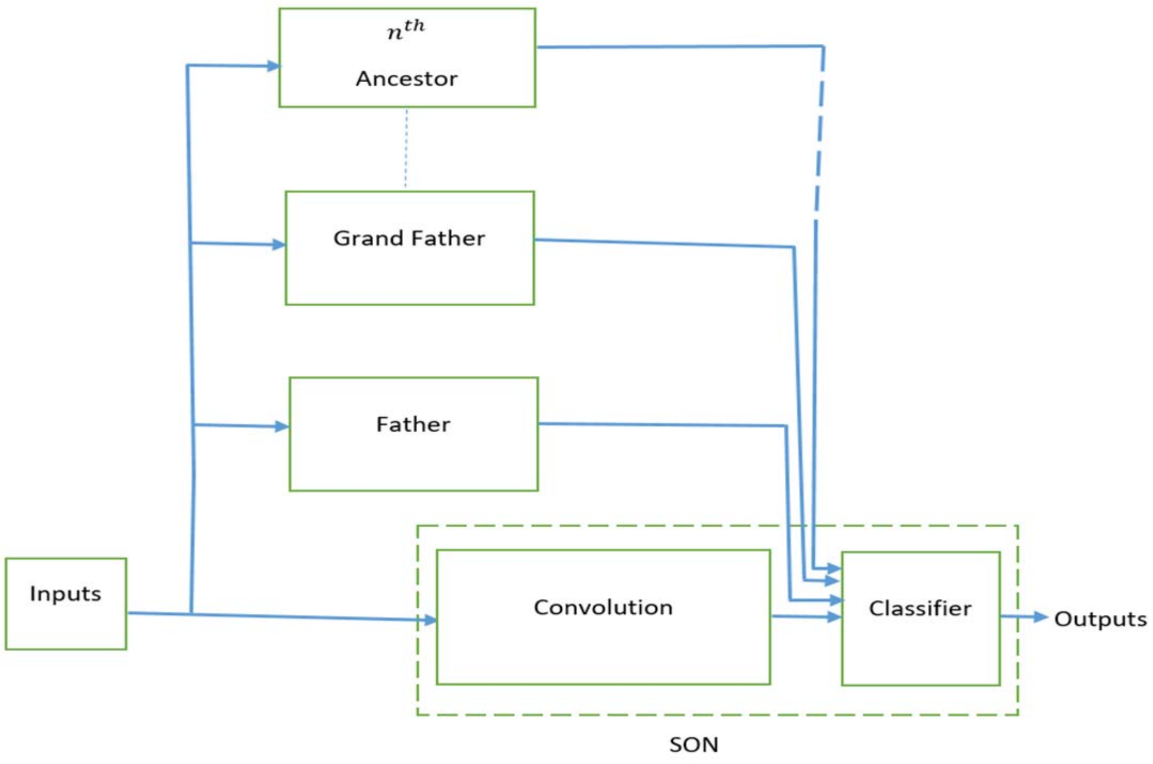

**Figure 3.23** Shows Intergenerational Interaction Neural Network Architecture(Source: Created by the author).

### 3.18 The proposed Father-Son Network: Self-organizing map Differential Convolutional neural network

The SOMdiffCNN integrates the concepts of Self-Organizing Maps (SOMs) and Differential Convolutional Neural Networks (Diff-CNNs) within a novel Father-Son Network framework. In this intergenerational interaction method, the SOM acts as the father, guiding the training of the son, DiffCNN model. The self-organizing map was initially pre-trained on the dataset, where it learns high-level feature representations through unsupervised clustering. These learned features are then used during each training step of the DiffCNN, forming the foundation of the son's learning process .Encoded inputs from the SOM are concatenated with the flattened output of the DiffCNN and subsequently passed through dense layers for classification.

### 3.18.1 The Proposed Method Architecture

The architecture of the proposed SOMdiffCNN typically includes an input layer (image), SOM class, differential convolutional class, DiffCNN model with Differential convolutional layers adapted in the first layer followed by convolutional layers and max-pooling layers (the output of the DiffConv layer is processed through convolutional layers, max-pooling, and dense layers), and then to the output layer. They are responsible for learning hierarchical representations of the input data, capturing spatial features at different levels of abstraction. The SOM layer performs the clustering and mapping of the learned features onto a low-dimensional grid.

In this method The SOM is initiated to extract features from the dataset it is a grid of neurons is utilized, in which every neuron represents a weight vector. During the preparation the trained SOM is used for each training step of the DiffCNN, also the SOM is used to encode inputs. (Self. SOM), and the encoded images are concatenated with the flattened output before being passed through the fully connected layers. Below is the proposed neural network; architectures

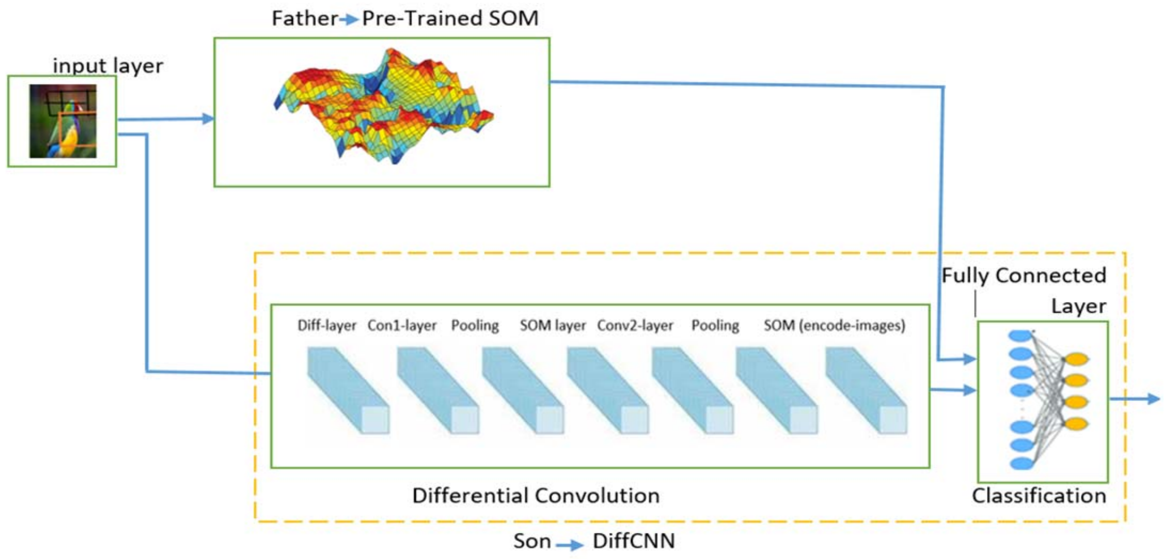

**Figure.3.24** Showing the proposed neural network architecture Diagram(Source: Created by the author).

#### 3.18.1.1 Input Layer Processes the image data.

#### 3.18.1.2 Differential convolutional layer

Adapts differential convolution techniques in the first layer of the DiffCNN, emphasizing spatial relationships in the data. The differential convolutional layers in the proposed method perform local receptive field operations on the input data .In the proposed methods the DiffCNN is used for classifying the data by using the trained SOM. It includes convolutional filters (or kernels) that slide across the input data (trained.som), extracting spatially local features. The non-linear activation function (ReLU) succeeds in each convolutional layer, a differential convolutional layer adapted in the model first layer, and pooling layers (e.g., in our case Max Pooling) to down-sample the feature Maps.

#### 3.18.1.3 SOM Layer

Extracts feature using a grid of neurons, each representing a weight vector. The SOM encodes input features into a high-dimensional space and performs competitive learning and clustering. It consists of a grid of neurons, where each neuron represents a weight vector. The weight vectors are adjusted during training to match the learned features from the differential convolutional layers. The SOM layer preserves the spatial relationships between the neurons, ensuring that similar features are mapped to nearby neurons.

#### 3.18.1.4 Convolutional and Max-Pooling Layers

Process the SOM-encoded input and extract hierarchical spatial features while down-sampling the feature maps.

#### 3.18.1.5 Fully Connected Layers and Output Layer:

Perform classification based on the concatenated SOM-encoded features and DiffCNN outputs. In this approach, the SOM serves a dual role: it provides pre-trained feature representations and dynamically encodes input features at each training step of the DiffCNN. The encoded outputs from the SOM are combined with the DiffCNN's features to enrich the learning process, enabling the DiffCNN to achieve faster convergence and better performance.

### 3.18.2 Training

The training process of the SOMdiffCNN in this thesis involves a two-step process. First, the features from the datasets are learned by the SOM and DiffCNN. Then, The SOM is trained to capture the dataset's structural relationships. The trained SOM is cast in each training step of the DiffCNN, also the SOM is used to encode inputs (Self. SOM), and The SOM-encoded features are integrated into the DiffCNN. The trained SOM guides the son model, concatenated with the flattened output before being passed through the fully connected layers. The training process also involves presenting input data, finding the best-matching unit (BMU) in the SOM layer, and updating the weights of the BMU and its neighboring neurons.,

### 3.18.3 Classification and Visualization

After training, the proposed SOMdiffCNN is used for the classification of the image datasets. Each input data point is matched to the neuron with the closest weight vector. Additionally, the low-dimensional grid of neurons in the SOM layer is visualized to gain insights into the learned features and their spatial organization. This intergenerational interaction framework allows the son model to benefit from the father model's prior knowledge, increasing performance and accelerating convergence. The proposed Father-Son Network architecture demonstrated superior accuracy and efficiency across multiple image datasets, showcasing its potential for broader applications in deep learning.

Figure 3.25 illustrates the architecture of the proposed SOMdiffCNN, highlighting the interplay between the father (SOM) and the son (DiffCNN) components.

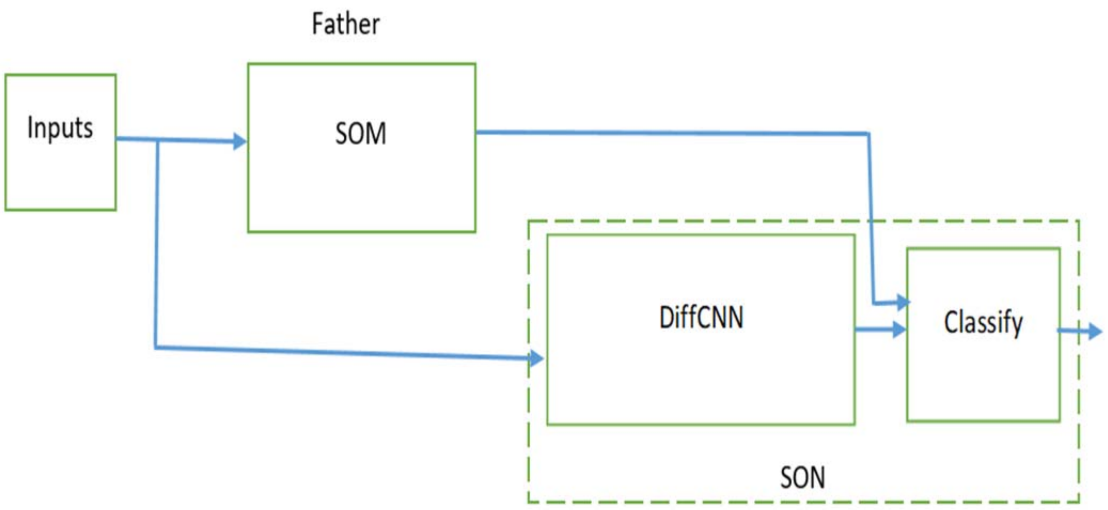

Figure 3.25 Shows the father-son Network Structure(Source: Created by the author).

Below displays the proposed neural network.SOMdiffCNN Model Algorithm

### 3.18.4 Algorithm2: the proposed neural network SOMdiffCNN Model Algorithm

```
Step 1: Import the required libraries
Step 2: Load and Preprocess Datasets
    Apply necessary transformations(normalization, convert image to tensor)
    Flatten and concatenate the data for SOM training.
Step 3: Initialize Self-Organizing Map Module:
    Define functions
        Distance Calculation: Use Euclidean distance to measure similarity
            $\text{Euclidean distance} = \sum_{i=1}^{n} (Xi + Wi)^2$
        Find Best Matching Unit (BMU): Identify the unit with the minimum distance
        Update the weights: update the weights of the BMU and its neighbors
            $wij(t+1) = wij(t) + \alpha(t).hij(t).(x(t) - wij(t))$
        Train the SOM
Step4:Call DiffConv Module:
    Call and implement the differentiable convolutional module (DiffConv)
Step5: DiffCNN Model Initialization
    Incorporate DiffConv layer followed by standard convolutional layers, and
```

```
        Encode inputs using the trained SOM and concatenate with the flattened output
        Create an Instance of DiffCNN with Trained SOM
            an instance of the defined model with specified output classes and the trained
        Define the model architecture
                Output=Fully Connected(Concat(SOM_Output,Flattened_Output))

        compile the model using
            Optimizer : Stochastic Gradient  descent  (SGD)
            Loss   function: cross-entropy  loss for classification tasks  .
    Step6:Train  the SOMdiffCNN network
        Train  The  network  using training data Conduct forward and backward passes
        Updates.
    Step 7: Evaluate  the  network
        evaluate The trained network  and print the results
    Step8: Visualize Results:
        Plot the performance of the SOM-based DiffCNN model Results
    End
```

### 3.19 The Structured Models

Table 3.1. The structured representation of the SOMdiffCNN Model

| Layer/Component | Type | Input Shape | Output Shape | Description |
|---|---|---|---|---|
| **Trained SOM** | SOM | (Batch Size, Features) | (Encoded Features) | Encodes input data using a trained Self-Organizing Map (SOM). |
| **DiffConv** | Custom DiffConv | (Batch_Size, Channels, Height, Width) | (Batch_Size, 5 * Channels, Height, Width) | Computes differential convolution to extract unique features. |
| **Conv1** | Conv2d | (Batch_Size, 5 * Channels, Height, Width) | (Batch_Size, 64, H-4, W-4) | Convolutional layer with 64 filters, kernel size 5x5, stride 1, and padding 0. |
| **Max Pooling** | MaxPool2d | (Batch_Size, 64, H-4, W-4) | (Batch_Size, 64, H/2, W/2) | Downsamples feature maps with kernel size 2x2. |
| **Flatten** | Reshape Operation | (Batch_Size, 64, H/2, W/2) | (Batch_Size, Flattened_Size) | Flattens the tensor into a 1D feature vector. |

Table 3.1. The structured representation of the SOMdiffCNN Model (continued)

| | | | | |
|---|---|---|---|---|
| **Concatenate** | Concatenate | (Flattened_Size) + (Encoded_Features) | (Combined_Features) | Combines the flattened CNN output with the encoded SOM features. |
| **Fully Connected Layer 1** | Linear | (Combined_Features) | (Batch_Size, 1024) | Fully connected layer with 1024 neurons. |
| **Fully Connected Layer 2** | Linear | (Batch_Size, 1024) | (Batch_Size, No Class) | Fully connected layer with classification (digits 0-9). |
| **Output** | Classification | (Batch_Size, Class) | (Batch_Size, No Class) | Final class probabilities for the dataset using CrossEntropyLoss. |

Table 3.2 The detailed structure of the DiffCNN model

| **Layer Name** | **Type** | **Input Shape** | **Output Shape** | **Description** |
|---|---|---|---|---|
| **diffconv** | DiffConv | [batch_size, 1, 28, 28] | [batch_size, 5, 28, 28] | Custom convolutional layer generating 5 feature maps. |
| **conv1** | Conv2d | [batch_size, 5, 28, 28] | [batch_size, 64, 24, 24] | Standard convolution with kernel size 5 and 64 filters. |
| **pool1** | MaxPool2d | [batch_size, 64, 24, 24] | [batch_size, 64, 12, 12] | Max pooling layer with kernel size 2. |

Table 3.2 the detailed structure of the DiffCNN model (continued)

| | | | | |
|---|---|---|---|---|
| **conv2** | Conv2d | [batch_size, 64, 12, 12] | [batch_size, 128, 8, 8] | Second convolution with kernel size 5 and 128 filters. |
| **pool2** | MaxPool2d | [batch_size, 128, 8, 8] | [batch_size, 128, 4, 4] | Max pooling layer with kernel size 2. |
| **fc1** | Linear (Fully Connected) | [batch_size, 128 * 4 * 4] | [batch_size, 1024] | Fully connected layer with 1024 output features. |
| **fc2** | Linear (Fully Connected) | [batch_size, 1024] | [batch_size, 10] | Fully connected layer with 10 output features (classes). |

Table 3.3 The Implemented CNN Structure.

| **Layer Name** | **Type** | **Input Shape** | **Output Shape** | **Description** |
|---|---|---|---|---|
| **conv1** | Conv2d | [batch_size, 1, 28, 28] | [batch_size, 32, 28, 28] | Convolution layer with 32 filters, kernel size 3, padding 1. |
| **relu1** | ReLU | [batch_size, 32, 28, 28] | [batch_size, 32, 28, 28] | ReLU activation. |
| **conv2** | Conv2d | [batch_size, 32, 28, 28] | [batch_size, 64, 28, 28] | Convolution layer with 64 filters, kernel size 3, padding 1. |
| **relu2** | ReLU | [batch_size, 64, 28, 28] | [batch_size, 64, 28, 28] | ReLU activation. |
| **pool1** | MaxPool2d | [batch_size, 64, 28, 28] | [batch_size, 64, 14, 14] | Max pooling with kernel size 2 and stride 2. |

Table 3.3 The Implemented CNN Structure.(continued)

| | | | | |
|---|---|---|---|---|
| **conv3** | Conv2d | [batch_size, 64, 14, 14] | [batch_size, 128, 14, 14] | Convolution layer with 128 filters, kernel size 3, padding 1. |
| **relu3** | ReLU | [batch_size, 128, 14, 14] | [batch_size, 128, 14, 14] | ReLU activation. |
| **conv4** | Conv2d | [batch_size, 128, 14, 14] | [batch_size, 128, 14, 14] | Convolution layer with 128 filters, kernel size 3, padding 1. |
| **relu4** | ReLU | [batch_size, 128, 14, 14] | [batch_size, 128, 14, 14] | ReLU activation. |
| **pool2** | MaxPool2d | [batch_size, 128, 14, 14] | [batch_size, 128, 7, 7] | Max pooling with kernel size 2 and stride 2. |
| **flatten** | Flatten | [batch_size, 128, 7, 7] | [batch_size, 128 * 7 * 7] | Flatten the feature maps for the fully connected layer. |
| **fc1** | Linear (Fully Connected) | [batch_size, 128 * 7 * 7] | [batch_size, 256] | Fully connected layer with 256 neurons. |
| **relu_fc1** | ReLU | [batch_size, 256] | [batch_size, 256] | ReLU activation. |
| **dropout1** | Dropout | [batch_size, 256] | [batch_size, 256] | Dropout layer with probability 0.5. |
| **fc2** | Linear (Fully Connected) | [batch_size, 256] | [batch_size, 128] | Fully connected layer with 128 neurons. |
| **relu_fc2** | ReLU | [batch_size, 128] | [batch_size, 128] | ReLU activation. |
| **dropout2** | Dropout | [batch_size, 128] | [batch_size, 128] | Dropout layer with probability 0.5. |
| **fc3** | Linear (Fully Connected) | [batch_size, 128] | [batch_size, 10] | Fully connected layer with 10 output neurons (classes). |

Table 3.4. The Architecture of Deep Convolutional Self-Organizing Map (DCSOM)

| DCSOM | **N-SOM** | **SOM-size** | **stride** | **patch/kernel** | **Auto encoder parameters** |
|---|---|---|---|---|---|
| layer1 | 81 | 20×20 | 1 | 16×16 | Encoder (Cnov2D,Relu , Maxpooling,Conv2D, Relu , Maxpooling) Decoder(Cnov2D, Relu,Upsampling,Conv2D,soft max,Upsampling ) |
| layer2 | 81 | 15× 15 | 1 | 8×8 | |
| layer3 | 36 | 10× 10 | 2 | 6×10 | |

## 3.20 Evaluation Metrics

In classification problems, the performance of a model is measured using a variety of evaluation metrics. These metrics evaluate the ability of the model to correctly classify data into predefined categories. In this research we employed two widely used metrics for assessment are: accuracy, and F1_score is the harmonic mean of Precision and Recall. It considers both false positives and false negatives, making it more reliable for imbalanced datasets. Below is an overview of key metrics used for assessing classification capability in this research:

### 3.20.1 Accuracy

Is the proportion of correct predictions out of the total predictions it is a straightforward metric often used when the dataset is balanced, meaning the classes are roughly equal in size. However, it can be misleading in cases of imbalanced datasets situations where one class significantly outweighs another (e.g., fraud detection). For example, predicting all outcomes as the majority class could yield high accuracy but fail to capture minority class performance (Raschka, 2018).

$$\text{Accutacy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where the TP is the True Positives, FN is False Negatives, TN is True Negatives and the FP is the False Positives.

### 3.20.2 F1-Score

F1-Score is the harmonic mean of Precision and Recall, making it particularly useful for imbalanced datasets. Precision focuses on the correctness of positive predictions, while Recall measures the ability to capture all true positive instances. The F1-Score balances these two, providing a reliable measure for tasks where the minority class is critical, such as detecting rare diseases (Han, Kamber, & Pei, 2011). For instance, a model with high Accuracy but low F1-Score might perform poorly on the minority class, emphasizing the importance of using F1-Score in relevant scenarios (Brownlee, 2020).

$$\mathrm{F1-Score} = 2 \times \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

Where the P is the Precision and the R is the recall.

### 3.20.3 Precision

The proportion of true positive predictions out of all positive predictions Precision measures the accuracy of positive predictions made by a classification model emphasizes correctness of positive predictions. It answers the question: Of all the instances predicted as positive, how many are actually positive (Raschka, 2018). Use Case: Precision is crucial when false positives are costly, such as in spam email detection, where wrongly classifying a legitimate email as spam can lead to loss of important information (Han, Kamber, & Pei, 2011).

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Where the TP True Positives is the Instances correctly classified as positive And the FP False Positives is the Instances incorrectly classified as positive.

### 3.20.4 Recall

Recall, also known as Sensitivity or the True Positive Rate, measures the ability of a classification model to identify all actual positive instances. It answers the question: Of all the actual positive cases, how many did the model correctly identify (Brownlee, 2020). **Use Case:** Recall is essential in scenarios where missing a positive case (false negatives) is costly, such as in medical diagnostics, where failing to identify a disease can lead to severe consequences (Han, Kamber, & Pei, 2011).

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Where True Positives (TP): Instances correctly classified as positive and False Negatives (FN): Instances incorrectly classified as negative (missed positives).

# 4. RESULTS AND DISCUSSIONS

The experiments and discussion of our proposed network are evaluated in this section using six different image datasets employed to assess the performance of the proposed method in all the experiments. The accuracy of the proposed methods was compared with that of other models. Such as the convolutional networks (CNN), differential convolutional networks (DiffCNN), and the deep convolutional self-organizing map (DCSOM) in all tests. All experiments aim to examine how the new models have improved and to assess the impact of changes in accuracy and performance. The primary goal is to investigate the effects of the new module. Hence, every experiment is being done to examine how the SOMdiffCNN has improved and to analyze the impact of changes in accuracy performance.

## 4.1 The datasets

The Photographer's Gallery's Data, Set, and Match program aims to share, visualize, and analyze modern image databases. Six image datasets are utilized to demonstrate the performance of the models: Fashion MNIST, Bird, MNIST, CIFAR-100, CIFAR-10, and STL-10 Datasets. The Bird Dataset contains high-resolution images of birds captured from multiple angles and in various environmental settings, includes 525 bird species, with 84,635 training images, 2,625 test images, and 2,625 validation images, where each image is a 224x224 color picture. The Fashion MNIST Dataset Each image is a 28x28 grayscale picture, categorized into 10 classes. Used for benchmarking computer vision algorithms. The MNIST Dataset is a set of handwritten digits, comprising 28x28 grayscale images of digits ranging from 0 to 9. Used for training models for handwritten digit recognition. The CIFAR-10 Dataset includes 60,000 color images of size 32x32, divided into 10 distinct classes for image classification tasks. The CIFAR-100 Dataset is an expanded version with 60,000 color images of size 32x32, categorized into 100 different classes for more complex classification tasks. STL-10 Dataset has 10,000 images of 96x96 color images and is used for developing and evaluating various semi-supervised and unsupervised learning methods.

To demonstrate the effectiveness of the techniques, 7 different experiment sets were carried out. These datasets are available to the public. Each dataset was normalized to have a zero mean and unit variance.
https://archive.ics.uci.edu/ml//index.phpdatasets,http://cs.joensuu.fi/sipu/datasets/ and **Hata! Köprü başvurusu geçerli değil.**www.kaggle.com/datasets/dhruvildave and their properties are shown in Table 1

.

**Table 4.1** Shows the dataset evaluation

| Data Sets Name | Attributes | Classes | Training Data | Test Data |
|---|---|---|---|---|
| MNIST | 64 | 10 | 60000 | 10000 |
| Fashion-mnist | 12 | 10 | 60000 | 10000 |
| Cifer10 | 16 | 10 | 50000 | 10000 |
| Bird | 3 | 525 | 84,635 | 2,625 |
| STL10 | 16 | 10 | 5000 | 8000 |
| Cifer100 | 16 | 100 | 500000 | 100000 |

Table 4.1. Shows the dataset evaluation.

## 4.2 The Experiments

### 4.2.1 The initial set

Comparing the accuracy of the SOMdiffCNN model across different datasets in this experiment, the proposed model assesses the resulting improvement of the Classification of six different image datasets. The model parameter values for all experiments are compiled in Table 1. The model consists of SOM, differential convolution, convolutional, and reshaping layers. The SOM was initially trained and then used in each training step in DiffCNN, The encoded inputs were then utilized by the SOM (Self. SOM) and concatenated with the encoded images before being passed through the dense layers. The results are displayed in Table 2. Overall, the model demonstrated a significant performance enhancement.

Table 4.2 Shown the hyper-parameter settings for the proposed model used in all experiments

| Parameter name | The values |
|---|---|
| Image Size | (28 × 28 × 1) pixels, (224 × 224 × 3) pixels, (32 × 32 × 3) pixels, And (96 × 96 × 3) pixels. |
| Number of Differential layers | 1 |
| batch size | 64 |
| SOM map dimension | 10×10 |
| Neighborhood radius(σi, σf ) | (1.0,0) |
| Number of epochs used in the learning | 50 |
| Learning rate | 0.001 |
| Batch normalization | - |
| kernel size | 3 |
| Map Size | 4-20 |
| Stride | 2 |

### 4.2.2 The Second series of Experiments

The effectiveness of our approach on the MNIST dataset was examined in comparison to the DiffCNN, CNN, and DCSOM models. The SOMdiffCNN was fine-tuned using a portion of the MNIST dataset (6000 training samples and 1000 testing samples) and the highest accuracy among all models was achieved by our model.

### 4.2.3 The Third set of Experiments

An examination is conducted on how well the suggested method performs on the Cifar10 dataset with normalization. The model begins with an input layer (an image of size 32 × 32 × 3 pixels), incorporating DiffConv Layer "subsequently followed by a convolutional layer by ReLU activation, max pooling, and fully connected layers." Subsequently, there are two fully connected (dense) layers (fc1 and fc2) with ReLU and softmax activation functions, respectively. The model was compiled using sparse categorical cross-entropy loss along with the stochastic gradient descent (SGD) optimizer. All the experiments in this paper The results, including accuracy, F1 score, and the loss values collected during training, are printed for evaluation, and compared to the DiffCNN, DCSOM, and CNN models.

### 4.2.4 The Fourth experiment set

The suggested method was compared with three models: DiffCNN, DCSOM, and CNN networks on Bird datasets. All compared structures had the same hyperparameter tuning. The method was defined using PyTorch indicating the Input layer: Accepts input data with shapes of (224, 224, and 3). It uses SOM, a differential convolutional layer. Followed by the standard convolutional layers (conv1, Conv2) with ReLU activation, a max-pooling layer, and layers that are fully connected. Subsequently, there are two fully connected (dense) layers (fc1 and fc2) with ReLU and softmax activation functions to produce output with 525 Classes of the softmax activation, suitable for multi-class classification tasks. The number 525 corresponds to the number of classes in the classification issue. The models are compiled using the categorical cross-entropy loss function, which is common for multi-class classification problems, and The Adam optimizer with a learning rate of 0.001 is utilized. Accuracy and F1 scores are then computed.

### 4.2.5 The fifth set of experiments

The correctness of the proposed approach on The STL10 dataset was investigated. STL-10 dataset was loaded using the Torchvision library and data loaders for training and testing were created. The model is defined as a PyTorch module that combines the SOM, differential convolutional layer (DiffConv layer), and DiffCNN with convolutional layers. Followed by the standard convolutional layers (conv1, conv2, etc.) with ReLU activation, a max-pooling layer, and dense layers. The network is trained with the specified loss function (cross-entropy) and optimizer

(Adam). Accuracy and the F1 score for both training and testing are calculated. Then, a comparison is done with three models DiffCNN, DCSOM, and CNN networks.

### 4.2.6 Experiment Sixth

The accuracy of the suggested Method was compared with three models: DiffCNN, DCSOM, and CNN models utilizing the Fashion MNIST dataset. The dataset is accessed via Keras.datasets.fashion_mnist. Additionally, the pixel values in the images were normalized to a range of 0 to 1. The data type of the images is modified to float32. The model is composed of SOM, DiffCNN with a DiffConvLayer as the main layer, double fully connected layers, max pooling, flattening, and a convolutional layer. The model was developed with the Adam optimizer, which results in a learning rate of 0.001 and sparse categorical cross-entropy loss. Then, 50 epochs were performed on the Fashion MNIST training. The accuracy and F1 scores are calculated for individual training and testing sets according to the labels assigned by SOM.

### 4.2.7 Experiment seven

The proposed method was tested on the Cifar100 dataset compared with the DiffCNN, DCSOM, and CNN models. The model consists of SOM and DiffCNN with a layer for input, diffConvLayer, flattening layers, convolutional layers, and max-pooling. The model is trained on the CIFAR-100 dataset using the stochastic gradient descent (SGD) optimizer and the sparse categorical cross-entropy loss for 50 epochs. Accuracy and F1 scores are calculated for the training and test set.

## 4.3 The Results

### 4.3.1 Experiment 1

The MNIST, Fashion-mnist, Cifar100, Birds, Cifar10, and STL10, Datasets were compared using the SOMdiffCNN method. Also, the accuracy was achieved by MNIST with an accuracy of 98.58%, While Fashion-mnist, Birds, STL10, Cifar100, and Cifar10, achieved accuracies of 96.531%, 87.49%, 86.9933%, 86.78%, and 81.657% respectively. The compression of our suggested approach with the six datasets is illustrated in Table 2.

### 4.3.2 The experiment 2

Comparisons of the accuracies of the proposed model with those of DiffCNN, CNN, and DCSOM, on the MNIST dataset. Accuracies of 98.01%, 95.73%, and 81.55% were achieved by DCSOM, DiffCNN, and CNN respectively our techniques demonstrated a very effective accuracy of 98.58%. The accuracy of 81.55% for CNN demonstrates poorer than our model (17.03%

improvement over CNN, 0.57% over DCSOM, and 2.85% over DiffCNN). The comparisons are presented in Table 3.

### 4.3.3 Experiment 3

In this test, the SOMdiffCNN, DiffCNN, CNN, and DCSOM, were compared with the Cifar10 dataset. Test accuracies of 81.657%, 80.88%, 79.26%, and 78.53% were achieved by SOMdiffCNN, DCSOM, CNN, and DiffCNN, respectively, According to the result of this dataset, Very effective accuracy was demonstrated by the SOMdiffCNN technique (3.12% improvement over DiffCNN, 0.77% over DCSOM, and 2.39% over CNN, ). The compression of our suggested approaches with the other techniques is shown in Table 4.

### 4.3.4 Experiment 4

In this test, the SOMdiffCNN, DiffCNN, CNN, and the DCSOM were compared on the Birds dataset. Our method achieved an accuracy of 87.49%. While DiffCNN, CNN, and DCSOM achieved 83.96%, 81.09%, and 76.46% respectively. In this dataset, our technique demonstrated a very effective test accuracy (3.53% improvement over DiffCNN, 6.4% over CNN, and 11.03% over DCSOM). Compression of the suggested methods with the other models is illustrated in Table 5.

### 4.3.5 Experiment 5

In this test, the SOMdiffCNN, DiffCNN, CNN, and DCSOM were compared on the STL10 dataset our proposed method Achieved 86.99% as its best accuracy on the STL10 dataset while DCSOM achieved 77.32%, CNN achieved 72.03% and DiffCNN achieved 85.16% accuracies. Our technique demonstrated (a 9.67% improvement over DCSOM, 1.83% over DiffCNN, and 14.96% over CNN). Table 6 shows the compression of our proposed approach with the other technique.

### 4.3.6 Experiment 6

In this experiment, our suggested method with DiffCNN, CNN, and DCSOM were comparedon the Fashion-mnist dataset. SOMdiffCNN achieved 96.53%, while DCSOM Achieved 93.39%, DiffCNN achieved 91.905%, and CNN achieved 88.56%, accuracies our technique demonstrated (3.19% improvement over DCSOM, 4.68% than DiffCNN and 8.02 % than CNN). Our proposed method achieved the best accuracy over all datasets. The compression is shown in Table 7.

### 4.3.7 Experiment 7

In this test, our proposed method, with DCSOM, CNN, and DiffCNN was compared with Cifar100 datasets. An accuracy of 86.78% was achieved byour method. While DCSOM Achieved 74.49%, DiffCNN achieved 75.06%, and CNN achieved 81.79%. Our technique demonstrated (a

12.29% improvement over DCSOM, 11.72% over DiffCNN, and 4.99% over CNN). The compression is presented in Table 8.

**Table 4.3** The accuracies of our proposed SOMdiffCNN comparisons on six different image datasets are presented.

| The model | The datasets | Train-Acc % | Test-Acc % | F1-score |
|---|---|---|---|---|
| SOMdiffCNN | MNIST | 98.9183 | 98.5816 | 98.5816 |
| | Cifar10 | 90.422 | 81.657 | 90.327 |
| | Birds | 88.65 | 87.49 | 87.49 |
| | STL10 | 88.023 | 86.9933 | 86.97 |
| | Cifar100 | 95.874 | 86.78 | 94.024 |
| | Fashion-mnist | 97.705 | 96.531 | 96.53 |

The experiment results show that the proposed methods demonstrate very effective accuracy in all the datasets. Achieved accuracy values of 98.58%, 96.53%, 87.49, 86.99%, 86.78%, and 81.65% in the MNIST, Fashion_MNIST, Birds, STL10, Cifer100, and Cifer10 datasets, respectively.

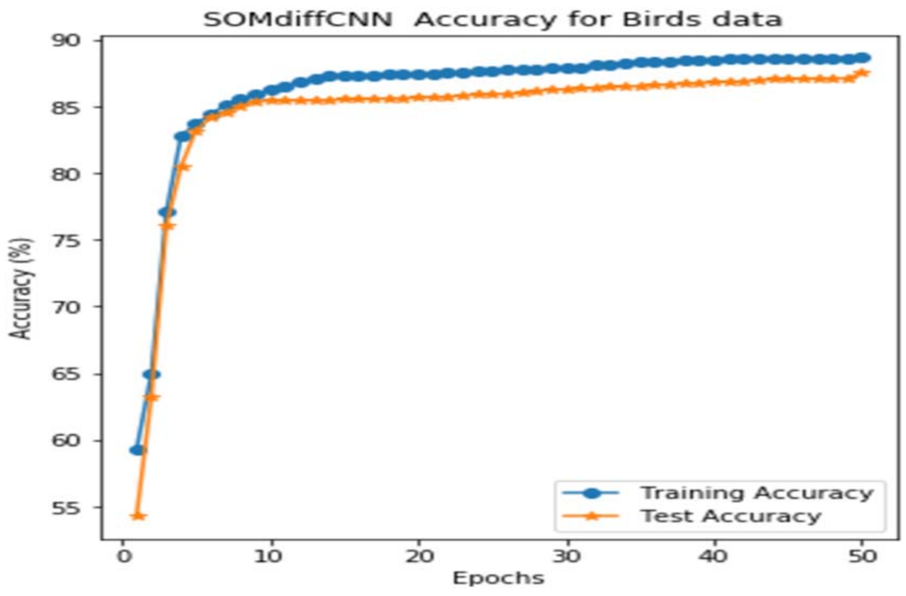

**Figure 4.1** SOMdiffCNN Model Accuracies for Birds Dataset.

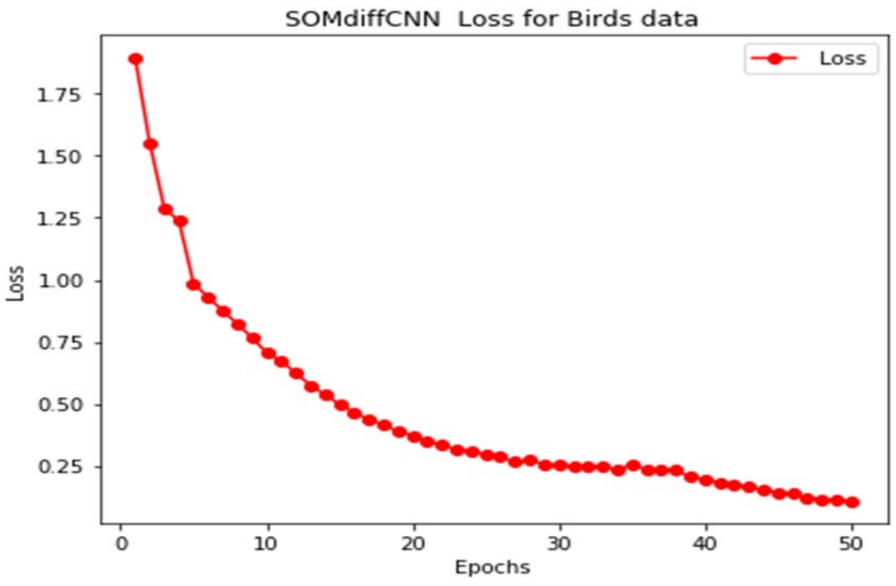

**Figure 4.2** SOMdiffCNN Model Training Loss for Birds dataset.

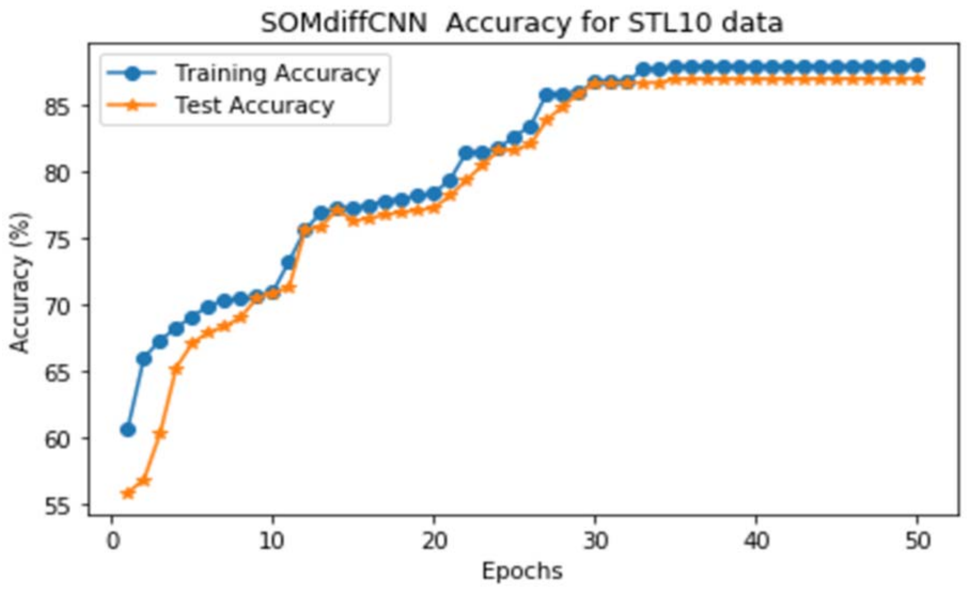

**Figure 4.3**. SOMdiffCNN Model Accuracies for STL10 Dataset.

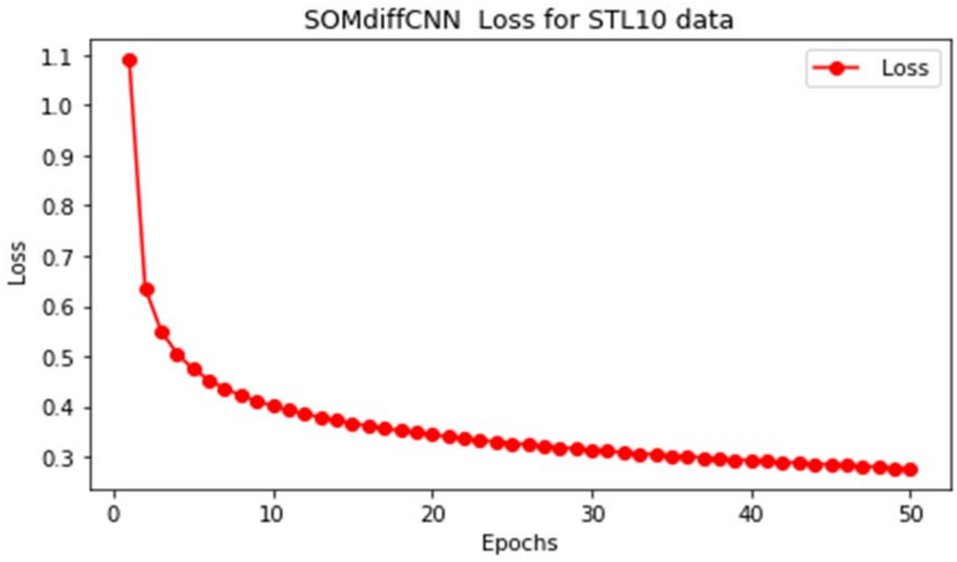

**Figure 4.4** SOMdiffCNN Model Accuracies for STL10 Datasets.

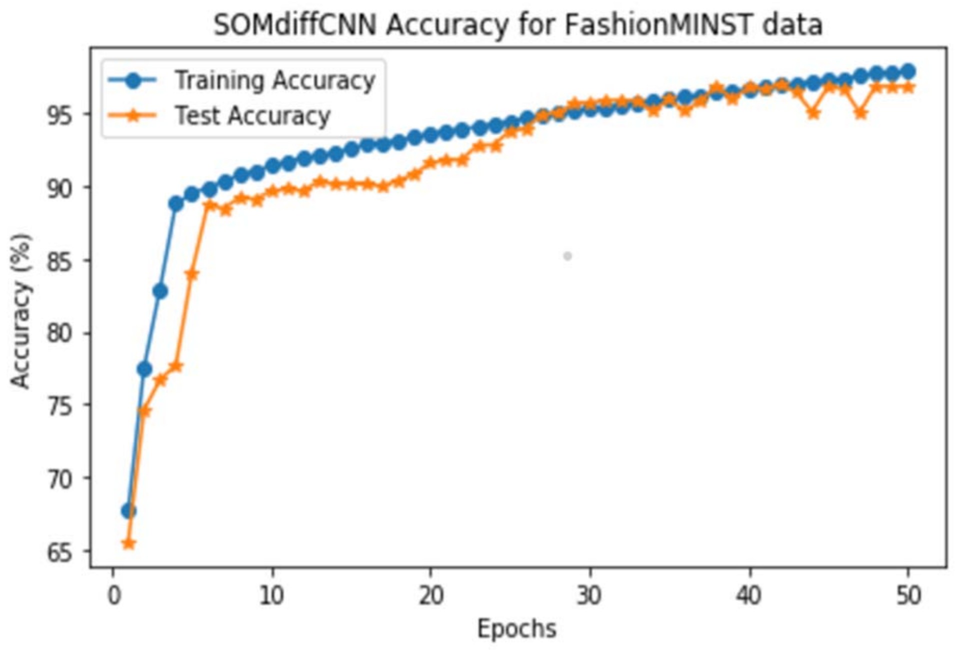

**Figure 4.5** SOMdiffCNN Model Accuracies for Fashion-mnist Dataset.

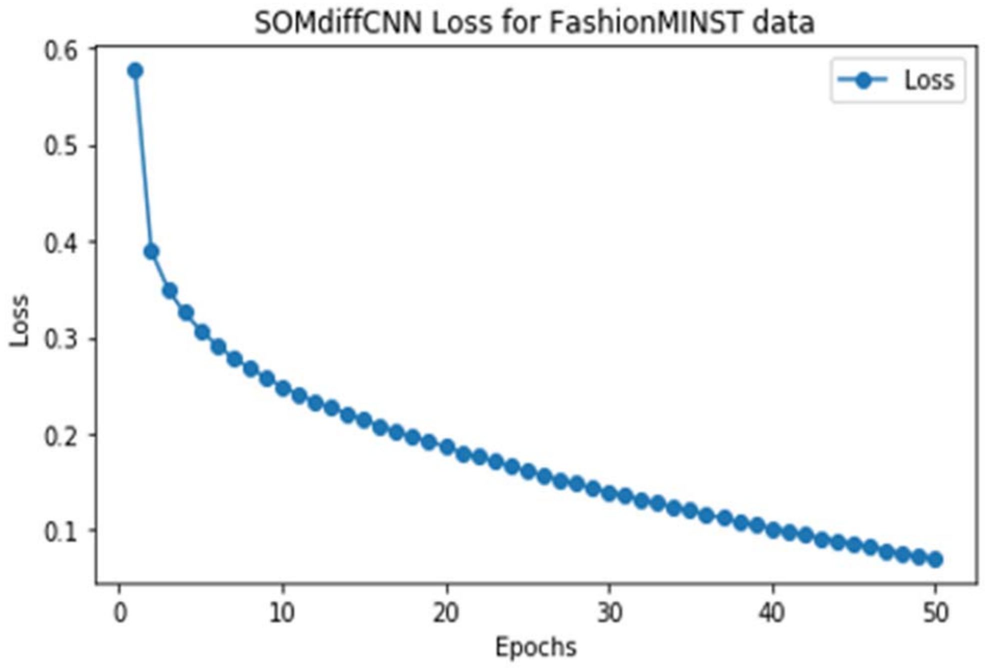

**Figure 4.6** SOMdiffCNN Model Training Loss for Fashion-mnist datasets.

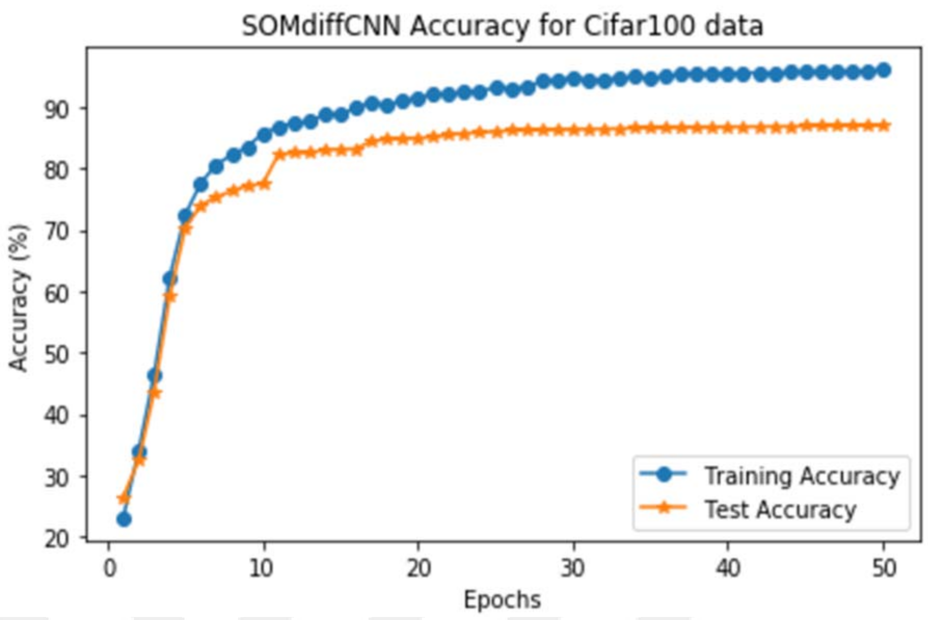

**Figure 4.7** SOMdiffCNN Model Accuracies for Cifar100 dataset.

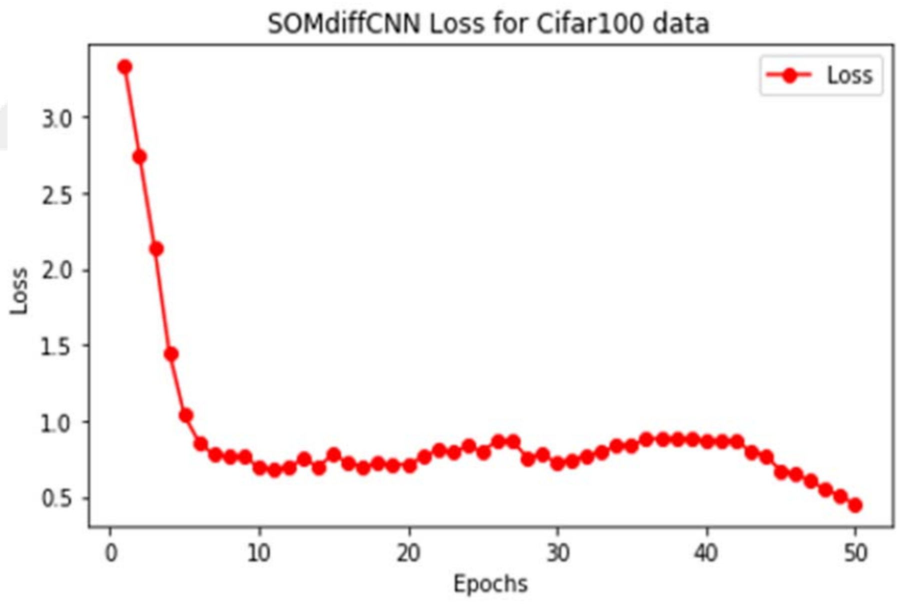

**Figure 4.8** SOMdiffCNN Model Training Loss for Cifar100 Dataset.

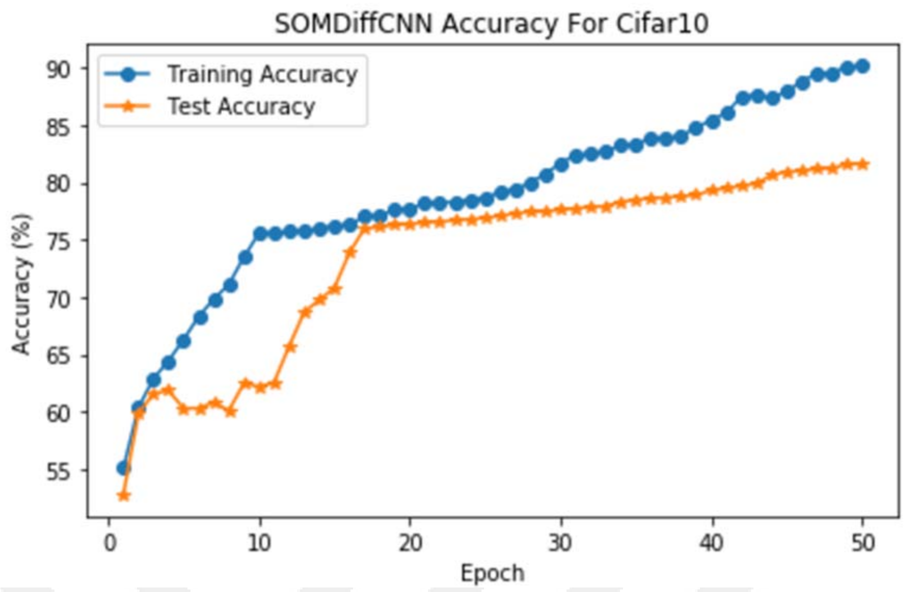

**Figure 4.9** SOMdiffCNN Model  Accuracies  for Cifar10 Dataset.

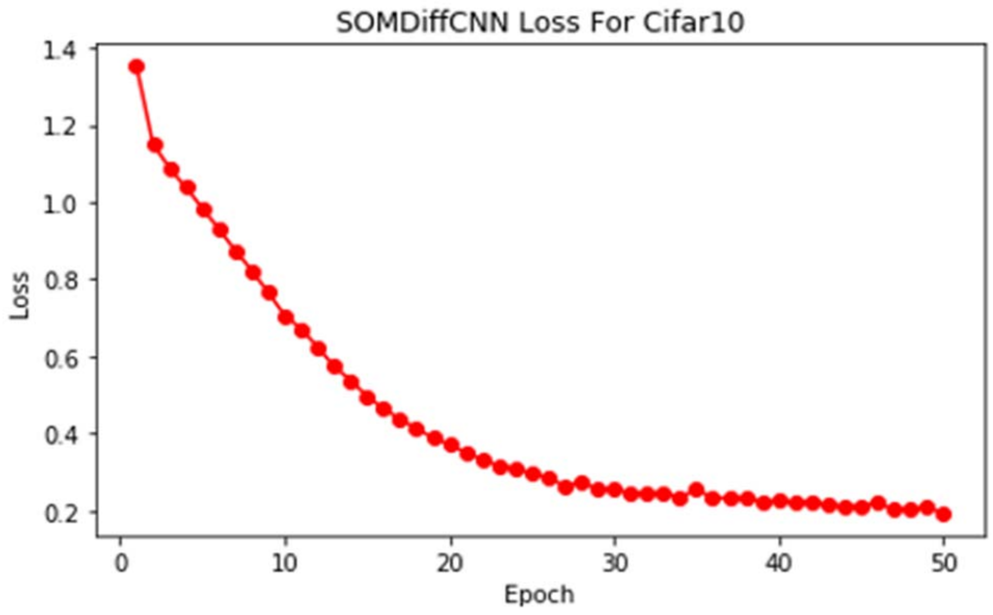

**Figure 4.10**  SOMdiffCNN Model  Training Loss  for Cifar10 Dataset.

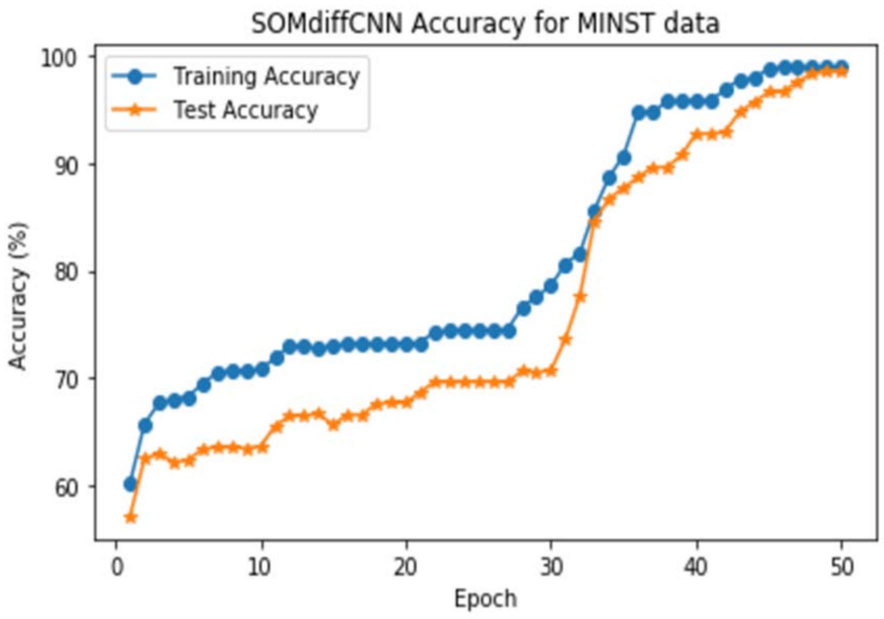

**Figure 4.11** SOMdiffCNN Model Accuracies for MNIST Dataset.

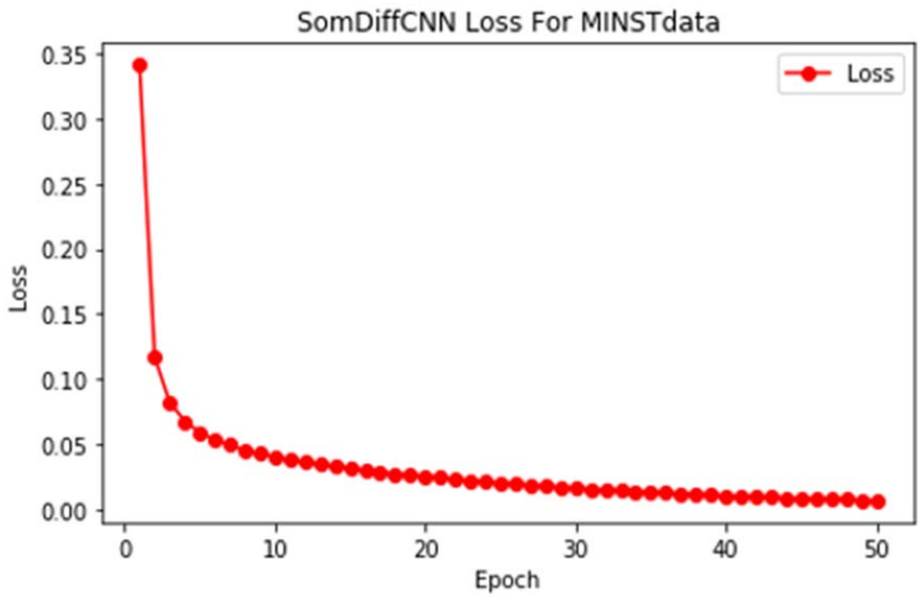

**Figure 4.12** SOMdiffCNN Model Training Loss for MNIST Dataset.

It is important to note that the term 'test accuracy' used in the plots throughout this thesis refers to the validation accuracy obtained during model development, not the final accuracy on the held-out test set .

**Table 4.4** The accuracy of our proposed method, comparisons with DiffCNN, CNN, and DCSOM on MNIST datasets are presented

| The datasets | The models | Train-Accuracy | Test-Accuracy | F1-Score | Loss Function |
|---|---|---|---|---|---|
| MNIST dataset | **SOMdiffCNN** | 98.9183 % | 98.5816 | 98.16% | 0.0065 |
| | DiffCNN | 95.91 % | 95.73 | 95.73% | 0.0437 |
| | CNN | 85.64 % | 81.55 | 81.55% | 0.0910 |
| | DCSOM | 98.11 % | 98.01 | 98.01% | 0.0810 |

**Table 4.5** displays the accuracies of our proposed method compared to DiffCNN, CNN, and DCSOM on CIFAR-10 datasets.

| The dataset | The models | Train-Accuracy | Test-Accuracy | F1-Score | Loss Function |
|---|---|---|---|---|---|
| Cifar10 datasets | **SOMdiffCNN** | 90.422 % | 81.657 | 81.657% | 0.1936 |
| | DiffCNN | 85.68 % | 78.53 | 78.53% | 0.3771 |
| | CNN | 79.763 % | 79.26 | 79.26% | 0.3790 |
| | DCSOM | 82.33 % | 80.05 | 80.13% | 0.3517 |

**Table 4.6** shown the accuracies of our proposed approach compared with DiffCNN,CNN, and DCSOM on Birds datasets.

| The dataset | The models | Train-Accuracy | Test-Accuracy | F1-Score | Loss Function |
|---|---|---|---|---|---|
| Birds datasets | **SOMdiffCNN** | 88.65 % | 87.49% | 87.49% | 0.1085 |
| | DiffCNN | 84.701 % | 83.96% | 83.63% | 0.1241 |
| | CNN | 81.19 % | 81.09% | 81.09% | 0.3491 |
| | DCSOM | 78.62 % | 76.46% | 76.44% | 0.2248 |

**Table 4.7** Presentations the accuracies of our proposed method compared to DiffCNN,CNN, And DCSOM on the STL10 datasets.

| The dataset | The models | Train-Accuracy | Test-Accuracy | F1-Score | Loss Function |
|---|---|---|---|---|---|
| STL10 | **SOMdiffCNN** | 89.023 % | 86.9933 | 86.99 | 0.2768 |
| | DiffCNN | 85.60% | 85.16% | 85.16% | 0,3443 |
| | CNN | 78.03% | 72.03% | 72.17% | 0.4224 |
| | DCSOM | 79.40 % | 77.32% | 77.22% | 0.3836 |

**Table 4.8** Confirmations the accuracies of our suggested Method comparisons with DiffCNN, CNN, and DCSOM on Fashion-mnist datasets are displayed.

| The dataset | The models | Train-Accuracy | Test-Accuracy | F1-Score | Loss Function |
|---|---|---|---|---|---|
| Fashion-MNIST dataset | **SOMdiffCNN** | 97.705 % | 96.58% | 96.58% | 0.0710 |
| | DiffCNN | 94.66% | 91.905% | 91.905% | 0.0814 |
| | CNN | 88.56% | 88.56% | 88.01% | 0.2900 |
| | DCSOM | 93.87 % | 93.398% | 93.39% | 0.1047 |

**Table 4.9** displays the accuracies of our suggested method compared with DiffCNN, CNN, and DCSOM on the Cifar100 datasets.

| The dataset | The models | Train-Accuracy | Test-Accuracy | F1-Score | Loss Function |
|---|---|---|---|---|---|
| Cifar100 | **SOMdiffCNN** | 95.874 % | 86.78% | 86.78% | 0.4025 |
| | DiffCNN | 89.76% | 75.06% | 75.06% | 0.3971 |
| | CNN | 79.33% | 81.79% | 78.87% | 0.4264 |
| | DCSOM | 90.03 % | 74.49% | 74.49% | 0.4925 |

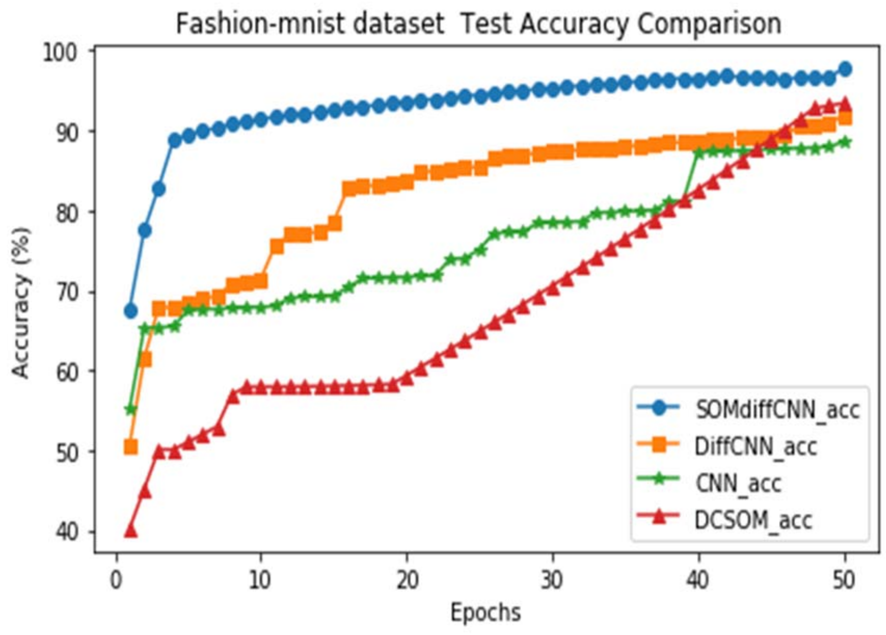

**Figure 4.13** Accuracy Comparison Results for Fashion-mnist Datasets.

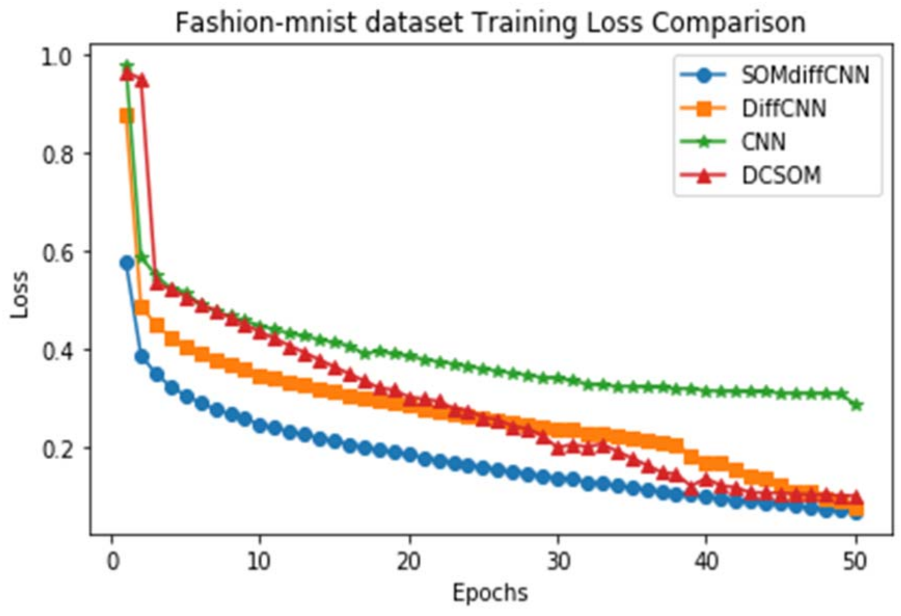

**Figure 4.14** Loss Comparison Results for Fashion-mnist Datasets.

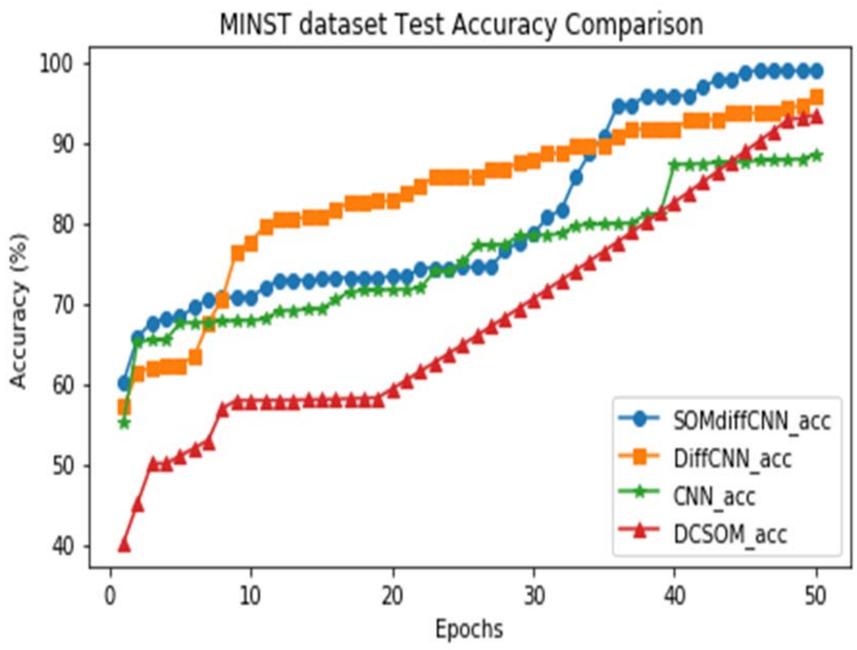

**Figure 4.15** Accuracy  Comparison Results for MNIST Datasets.

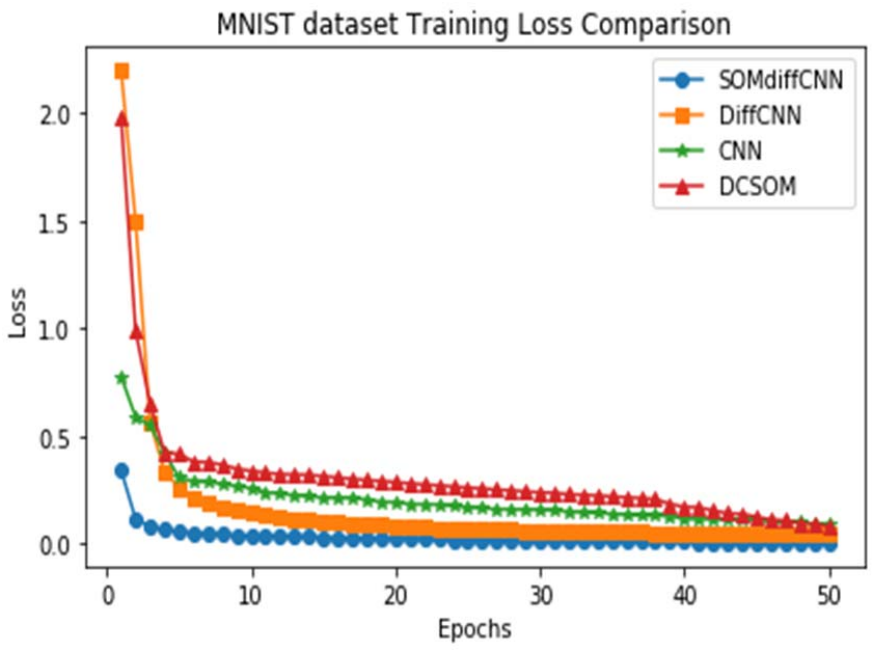

**Figure 4.16** Loss Comparison Results for MNIST Datasets

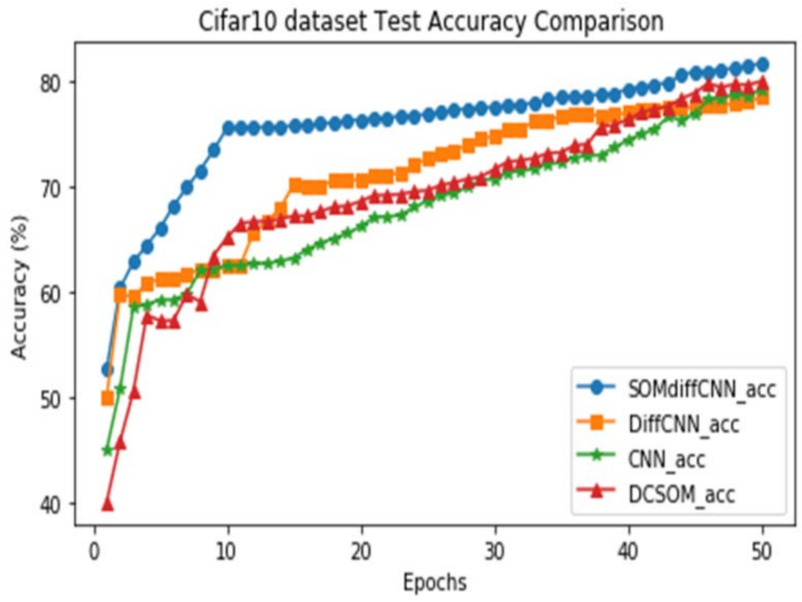

**Figure 4.17** Accuracy Comparison Results for Cifar10 Datasets.

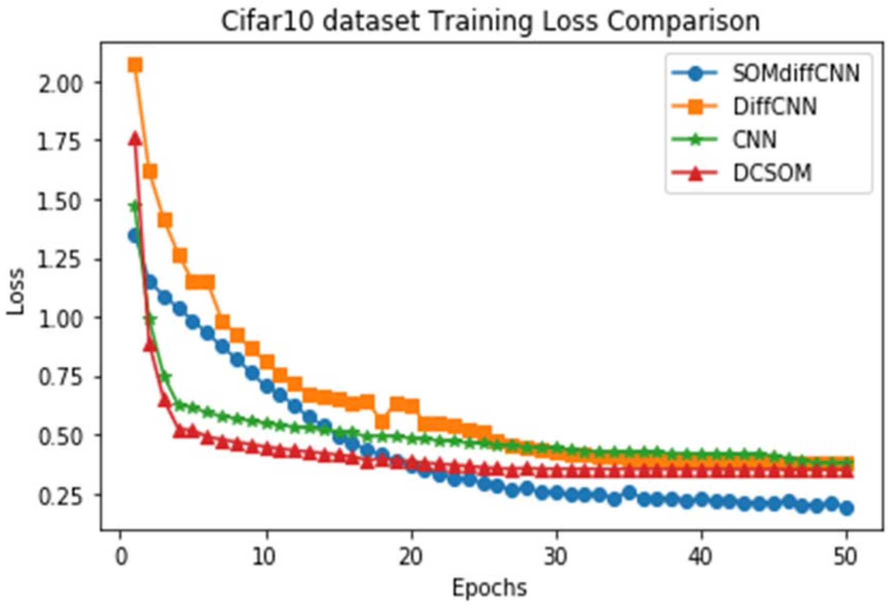

**Figure 4.18** Loss Comparison Results for Cifar10 Datasets.

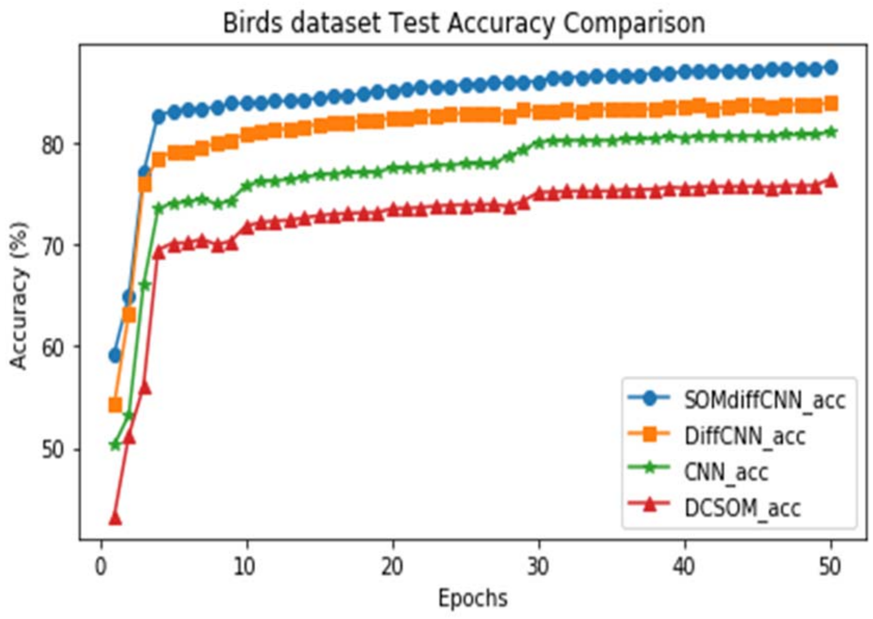

**Figure 4.19** Accuracy Comparison  Results for Birds Datasets.

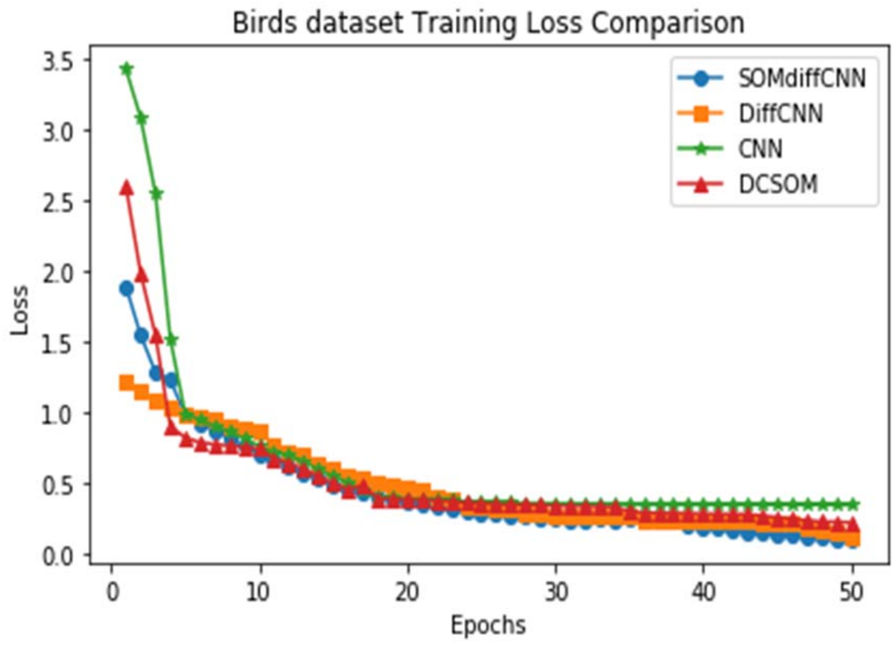

**Figure 4.20** Loss Comparison Results for Birds dataset.

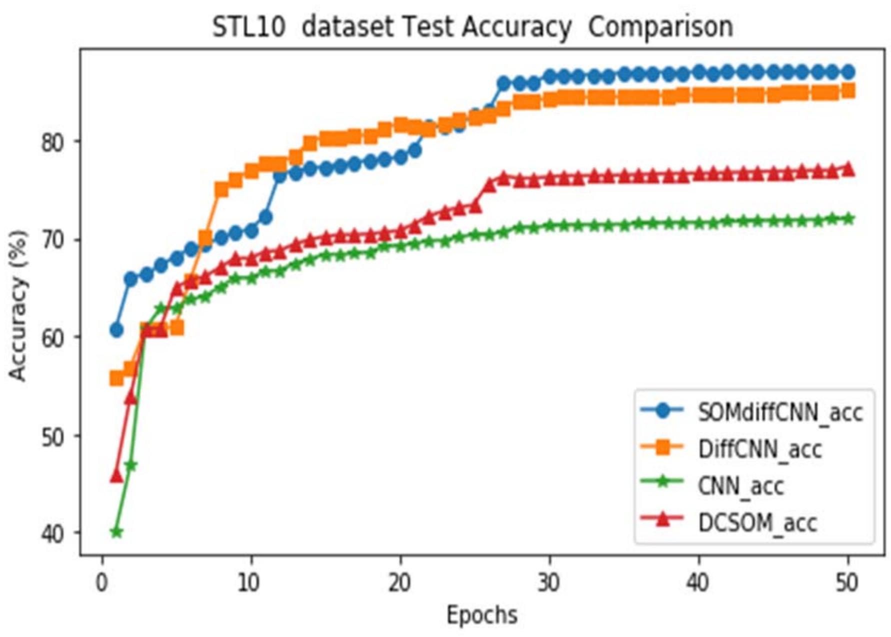

**Figure 4.21.** Accuracy Comparison Results for STL10 Datasets.

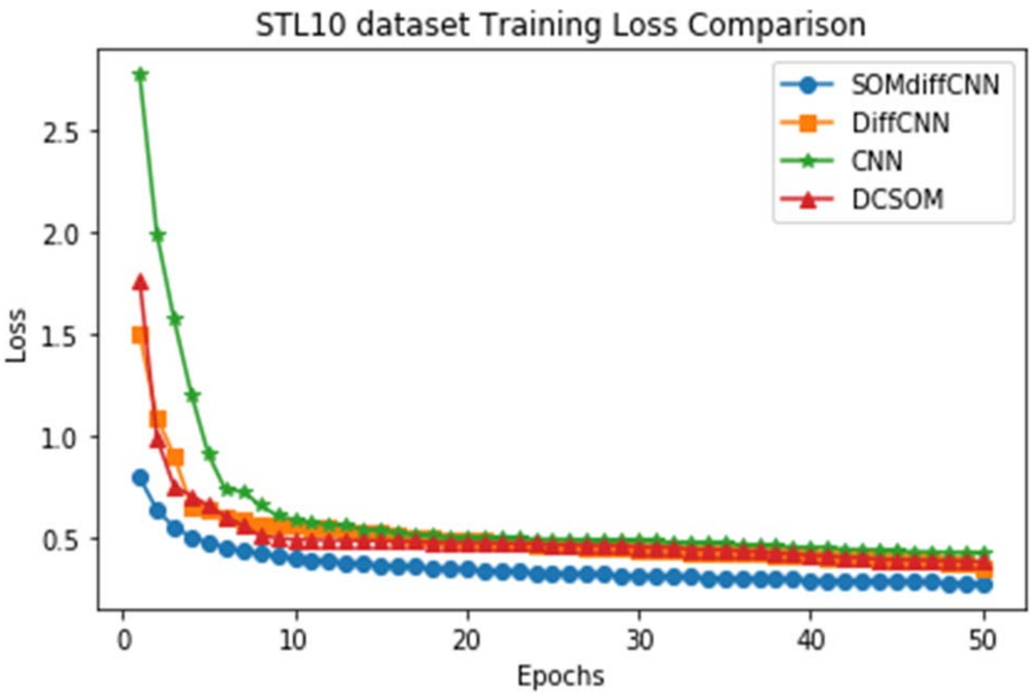

**Figure 4.22.** loss Comparison Results for STL10 Datasets.

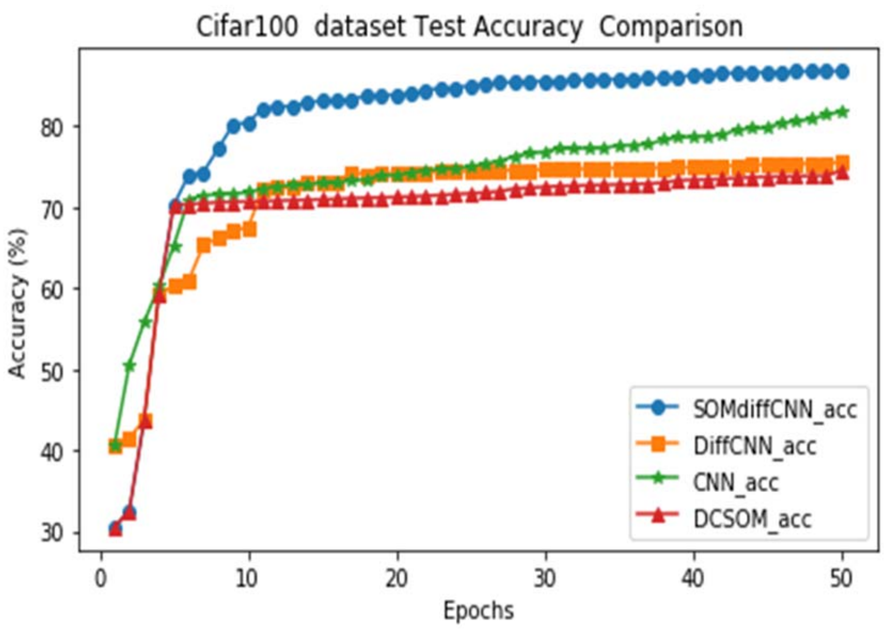

**Figure 4.23**. Accuracy Comparison Results for Cifar100 Datasets.

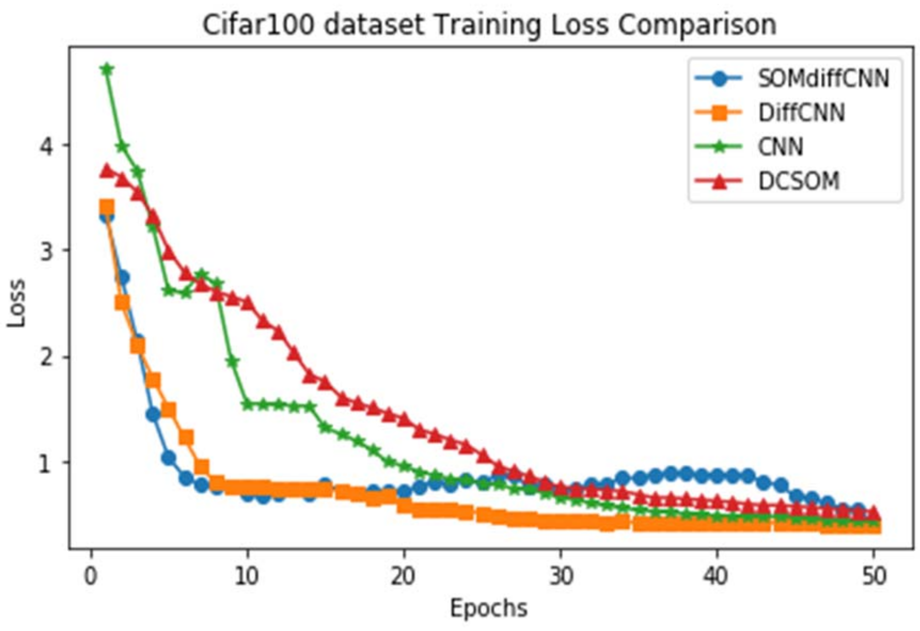

**Figure 4.24.** loss Comparison Results for Cifar100 Datasets.

Within this work, we suggested a novel Intergenerational Interaction Type of Neural Network Self-Organizing Map Differential Convolutional Neural Network for classifying image datasets.

A new perspective on the Differential Convolutional Neural Network has been considered, and the hypothesis behind the method is that a building father's presence enables the son to succeed quicker than others. Self-organizing map stands for the building father it is pre-trained by experience before training his son. Son is the Differential convolutional neural network where in this case is used. Since this Enhancement increases both learning and classification speed, it leads to wider application possibilities for the proposed topology. The proposed architecture is distinct and different from further SOM-based DL. Tested on six different datasets, it demonstrated very effective accuracy in all the datasets.

In the First test, we compared the DiffCNN, CNN, and, DCSOM, with the MINST dataset DCSOM, DiffCNN, and, CNN achieved 98.01%, 95.73%, and 81.55%, respectively our technique demonstrated a very effective accuracy. The accuracy of 81.55 % for CNN demonstrates poor data classification than our Model.

In the Second test, we compared the SOMdiffCNN, DiffCNN, CNN, and DCSOM, with the Ciar10 dataset SOMdiffCNN, DCSOM, CNN, and DiffCNN achieved test accuracies of 81.657%, 80.88, 79.26%, and 78.53 respectively. According to the results of this datasets our technique demonstrates a very effective Classification accuracy (3.12% improvement over DiffCNN, 0.77% than DCSOM, and 2.39 over DCNN).

In the Third test, the SOMdiffCNN, DiffCNN, CNN, and the DCSOM were compared on the Birds dataset. Our method achieved an accuracy of 87.49%. While DiffCNN, CNN, and DCSOM achieved 83.96%, 81.09%, and 76.46% respectively. In this dataset, our technique demonstrated a very effective test accuracy (3.53% improvement over DiffCNN, 6.4% over CNN, and 11.03% over DCSOM).

In the Fourth test, the SOMdiffCNN, DiffCNN, CNN, and DCSOM were compared on the STL10 dataset our proposed method Achieved 86.99% as its best accuracyon the STL10 dataset while DCSOM achieved 77.32%, CNN achieved 72.03% and DiffCNN achieved 85.16% accuracies. Our technique demonstrated (a 9.67% improvement over DCSOM, 1.83% over DiffCNN, and 14.96% over CNN).

In the Fives test, our suggested method with DiffCNN, CNN, and DCSOM were compared on the Fashion-mnist dataset. SOMdiffCNN achieved 96.53%, while DCSOM Achieved 93.39%, DiffCNN achieved 91.905%, and CNN achieved 88.56%, accuracies our technique demonstrated (3.19% improvement over DCSOM, 4.68% than DiffCNN and 8.02 % than CNN).

In the sixth test our proposed method, with DCSOM, CNN, and DiffCNN was compared with Cifar100 datasets. An accuracy of 86.78% was achieved by our method. While DCSOM Achieved 74.49%, DiffCNN achieved 75.06%, and CNN achieved 81.79%. Our technique demonstrated (a 12.29% improvement over DCSOM, 11.72% over DiffCNN, and 4.99% over CNN).

The experiments showed that the proposed method demonstrated very effective accuracy in all the datasets. The accuracy values achieved by 98.58%, 96.53%, 87.49%, 86.99%, 86.78 and 81.65% in the MNIST, Fashion_MNIST, Birds, STL10, Cifer100, and Cifar10, datasets, respectively, surpassing the latest advancements performance.

**Table 4.10** Optimized parameters for the models (CNN, DCSOM, SOMdiffCNN, and DiffCNN).

| Properties | DCNN | DCSOM | SOMdiffCNN | DiffCNN |
|---|---|---|---|---|
| Learning late | 0.001 | 0.001 | 0.001 | 0,001 |
| Epochs | 50 | 50 | 50 | 50 |
| Bach Size | 64 | 64 | 64 | 64 |
| Number of Filter | 512 | 512 | 512 | - |
| Filter Size | 3 | 3 | 3 | - |
| pooling | Maxpooling2D | Maxpooling2D, Upsampling2D | Maxpooling2D, Upsampling2D | Maxpooling2D |
| Differential layer | - | - | yes | yes |
| Euclidean Distance | - | yes | yes | yes |
| Manhattan distance | - | - | - | - |
| Activation Function | Relu , Softmax , sigmoid | | | |
| Loss | Binary Cross entropy | | | |
| Optimizer | Adam , SGD | | | |

Table 4.10. Shows the Optimized parameters for the models (CNN, DCSOM, Diff-CSOM, and DiffCNN).

## 5. CONCLUSIONS

This research introduced a novel hybrid neural network framework, termed Intergenerational Interaction Neural Networks (IINNs), which integrates the principles of Self-Organizing Maps (SOMs) and Differential Convolutional Neural Networks (DiffCNNs). Inspired by the philosophy of generational guidance. The proposed method leverages the synergy between a Self-Organizing Map (SOM), representing the guiding father, and a Differential Convolutional Neural Network (DiffCNN), symbolizing the son The hypothesis behind this method is that "The presence of a guiding father enables the son  to succeed quicker and better than the others" to enhance the faster convergence and improve the performance. Unlike traditional teacher-student frameworks the proposed approach involves pre-trained father or ancestor models working together with the son model during both the training and application phases of the son network.

The SOMdiffCNN represents a unique Father-Son Network**,**By leveraging the SOM's unsupervised feature extraction and Encoding capabilities, the DiffCNN benefits from pre-learned structural representations, which are dynamically integrated during each training step. This intergenerational approach significantly reduces the complexity of the convolutional layers in the DiffCNN, while simultaneously improving convergence speed and overall classification accuracy.

The SOM was pre-trained independently before integrating with the DiffCNN, where its output was concatenated with encoded inputs. This hybrid architecture demonstrated superior adaptability and feature representation capabilities, as evidenced by its application to six diverse image datasets.

The effectiveness of the SOMdiffCNN was validated across six image datasets MNIST, Fashion-MNIST, Birds, STL10, CIFAR10, and CIFAR100. Resulting in classification accuracy values of 98.58%, 96.53%, 87.49%, 86.99%, 86.78%, and 81.65% respectively. Moreover, the model demonstrated significant performance improvements within the first 10 training epochs. Experimental results indicate notable enhancements in both training and testing accuracy. For the MNIST dataset, the model improved 10.62% in training accuracy and 6.53% in testing accuracy. Similarly, for the Fashion-MNIST, CIFAR-10, and CIFAR-100 datasets, the model recorded accuracy improvements of 23.72% / 24.16%, 20.48% / 9.42%, and 62.48% / 50.92%, respectively. Additionally, notable performance gains were observed on the Birds and STL-10 datasets, with accuracy improvements of 26.92% / 31.14% and 10.21% / 15.03%, respectively. The proposed model achieves faster convergence and significant performance improvements across multiple datasets within the first 10 training epochs. The experimental results underscore the model's ability to achieve faster convergence up to 84%, reaching 85% accuracy on more complex datasets, such as CIFAR-10, Birds, and CIFAR-100 within the first 7 to 10 epochs.  Maintaining strong performance across simpler datasets like FashionMNIST, where it reached 90% accuracy by the 7th epoch, resulting in a 74% faster convergence. The proposed model achieved state-of-the-art accuracy rates, surpassing traditional CNN, DiffCNN, and DCSOM models. These results highlight the ability of the SOMdiffCNN to balance computational efficiency and predictive performance, particularly in resource-constrained environments.

The findings validate the hypothesis that leveraging generational interactions within neural networks fosters a more efficient and effective learning paradigm. The intergenerational approach not only advances the theoretical understanding of hybrid neural networks but also offers a practical solution for enhancing model performance on challenging classification tasks.

This work contributes to the growing field of deep learning by presenting an innovative hybrid architecture that combines unsupervised and supervised learning principles.The Father-Son Network paradigm emphasizes the importance of leveraging intergenerational knowledge transfer to build more efficient and adaptive neural networks. Future research may explore the scalability of this framework, its applicability to other domains, such as natural language processing or time-series data, investigating the inclusion of deeper ancestral structures and the integration of additional hierarchical generational layers to further enhance performance. The developed topology is applicable to medical image recognition, image recognition in automotive industry with better performance advantages and real time image recognition tasks such as automatic driving systems with much faster learning speed.

By introducing this work the aim is to bridge the gap between traditional hierarchical learning methods and a more collaborative, intergenerational approach, advancing both the conceptual framework and practical applications of deep learning.

This study sets a promising foundation for next-generation neural network architectures, highlighting the potential of biologically inspired methodologies in advancing artificial intelligence.

# REFERENCES

Alexander Amini, and Ava Soleimany. (2020). *Introduction to Deep Learning | Electrical Engineering and Computer Science*. (n.d.). MIT OpenCourseWare. Retrieved June 17, 2023, from https://ocw.mit.edu/courses/6-s191-introduction-to-deep-learning-january-iap-2020/

Aly, S., Shimada, A., Tsuruta, N., & Taniguchi, R. (2010, August 1). Robust Face Recognition Using Multiple Self-Organized Gabor Features and Local Similarity Matching. IEEE Xplore. https://doi.org/10.1109/ICPR.2010.71

Aly, S., & Almotairi, S. (2020). Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition. IEEE Access, 8, 107035–107045. https://doi.org/10.1109/access.2020.3000829

Akdemir, D., & Jannink, J.-L. (2014). Ensemble learning with trees and rules: Supervised, semi-supervised, and unsupervised. *Intelligent Data Analysis*, *18*(5), 857–872. https://doi.org/10.3233/ida-140672

A Deep Learning Framework for Predicting Response to Therapy in Cancer. (2019). *Cell Reports*, *29*(11), 3367-3373.e4. https://doi.org/10.1016/j.celrep.2019.11.017

Agarwal, P. (2016). Machine Learning Toolbox. *Machine Learning and Applications: An International Journal*, *3*(3), 25–34. https://doi.org/10.5121/mlaij.2016.3303

Altaf, F., Islam, S. M. S., Akhtar, N., & Janjua, N. K. (2019). Going Deep in Medical Image Analysis: Concepts, Methods, Challenges, and Future Directions. *IEEE Access*, *7*, 99540–99572. https://doi.org/10.1109/access.2019.2929365

Aaron, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). *WaveNet: A Generative Model for Raw Audio*. ArXiv.org. https://arxiv.org/abs/1609.03499

Alexander Amini, and Ava Soleimany.(2020).A SURVEY ON DEEP LEARNING TECHNIQUES. (2020). *Strad Research*, *7*(8). https://doi.org/10.37896/sr7.8/037

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2018.00636

Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., & Mougiakakou, S. (2016). Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network. *IEEE Transactions on Medical Imaging*, *35*(5), 1207–1216. https://doi.org/10.1109/TMI.2016.2535865

About Python. (n.d.). Www.pythoninstitute.org. https://www.pythoninstitute.org/about-python

Aly, S., & Aly, W. (2020). DeepArSLR: A Novel Signer-Independent Deep Learning Framework for Isolated Arabic Sign Language Gestures Recognition. IEEE Access, 8, 83199–83212. https://doi.org/10.1109/access.2020.2990699

Aly, S., & Almotairi, S. (2020). Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition. IEEE Access, 8, 107035–107045. https://doi.org/10.1109/access.2020.3000829

Aly, S., Tsuruta, N., Taniguchi, R.-I., & Shimada, A. (2008, June 1). Visual feature extraction using variable map-dimension Hypercolumn Model. IEEE Xplore. https://doi.org/10.1109/IJCNN.2008.4633896

Aly, S., & Tsuruta, N. (2009). On Face Recognition using Hierarchical Self-organized Gabor Features. Retrieved September 21, 2023, from https://www.mva-org.jp/Proceedings/2009CD/papers/13-23.pdf

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A. S., & Asari, V. K. (2019). A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, *8*(3), 292. https://doi.org/10.3390/electronics8030292

Abdi, A. H., Luong, C., Tsang, T., Allan, G., Nouranian, S., Jue, J., Hawley, D., Fleming, S., Gin, K., Swift, J., Rohling, R., & Abolmaesumi, P. (2017). Automatic Quality Assessment of Echocardiograms Using Convolutional Neural Networks: Feasibility on the Apical Four-Chamber View. *IEEE Transactions on Medical Imaging*, *36*(6), 1221–1230. https://doi.org/10.1109/tmi.2017.2690836

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A. K., Dean, J., Devin, M., Sanjay Ghemawat, Irving, G., Isard, M., Manjunath Kudlur, Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P. G., Vasudevan, V. K., Warden, P., & Wicke, M. (2016). TensorFlow: a system for large-scale machine learning. *Operating Systems Design and Implementation*, 265–283. https://doi.org/10.5555/3026877.3026899

Akinduko, A. A., & Mirkes, E. M. (2012). Initialization of Self-Organizing Maps: Principal Components Versus Random Initialization. A Case Study. *ArXiv (Cornell University)*.

Abd El Kader, I., Xu, G., Shuai, Z., Saminu, S., Javaid, I., & Salim Ahmad, I. (2021). Differential Deep Convolutional Neural Network Model for Brain Tumor Classification. *Brain Sciences*, *11*(3), 352. https://doi.org/10.3390/brainsci11030352

Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A., & Tan, R. S. (2017). A deep convolutional neural network model to classify heartbeats. *Computers in Biology and Medicine*, *89*, 389–396. https://doi.org/10.1016/j.compbiomed.2017.08.022

Appiah, K., Hunter, A., Dickinson, P., & Meng, H. (2012). Implementation and Applications of Tri-State Self-Organizing Maps on FPGA. *IEEE Transactions on Circuits and Systems for Video Technology*, *22*(8), 1150–1160. https://doi.org/10.1109/tcsvt.2012.2197077

Author Rahul Kumar. (n.d.). *Machine learning quick reference : quick and essential machine learning hacks for training smart data models*. Packt Uuuu-Uuuu.

Anderson, J. A. (1995). *An Introduction to Neural Networks*. MIT Press.

Artificial intelligence abstracts. (1987). *Artificial Intelligence*, *32*(3), 414–415. https://doi.org/10.1016/0004-3702(87)90098-1

A neural unsupervised learning technique. (1988). *Neural Networks*, *1*, 69. https://doi.org/10.1016/0893-6080(88)90108-6

Abadi, M. (2016). TensorFlow: learning functions at scale. ACM SIGPLAN Notices, 51(9), 1–1. https://doi.org/10.1145/3022670.2976746

Bondarenko, A. N., & Katsuk, A. V. (2007, April 1). *Application of Self-Organization Maps to the Biomedical Images Classification*. IEEE Xplore. https://doi.org/10.1109/SIBCON.2007.371312

Box, P., Van Der Maaten, L., Postma, E., & Van Den Herik, J. (2009). Tilburg centre for Creative Computing Dimensionality Reduction: A Comparative Review Dimensionality Reduction: A Comparative Review. http://lvdmaaten.github.io/publications/papers/TR_Dimensionality_Reduction_Review_2009.pdf

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2), 281-305.

Brady, J. M. (2020). Artificial intelligence and natural man. *Artificial Intelligence*, *11*(3), 267–269. https://doi.org/10.1016/0004-3702(78)90003-6

Bieniecki, W., Grabowski, S., & Rozenberg, W. (2007). Image Preprocessing for Improving OCR Accuracy. 2007 International Conference on Perspective Technologies and Methods in MEMS Design. https://doi.org/10.1109/memstech.2007.4283429

Bill Wilson.(2010). Self-organisation Notes, www.cse.unsw.edu.au/~billw/cs9444/selforganising-10-4up.pdf

Bergstra, J., Olivier Breuleux, Bastien, F., Lamblin, P., Razvan Pascanu, Desjardins, G., Turian, J., Warde-Farley, D., & Yoshua Bengio. (2010). *Theano: A CPU and GPU Math Compiler in Python*. https://doi.org/10.25080/majora-92bf1922-003

Brownlee, J. (2021, April 18). A Gentle Introduction to Ensemble Learning Algorithms. Machine Learning Mastery. https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/

Becker, H. (2001). Computing with words and machine learning in medical diagnostics. *Information Sciences*, *134*(1-4), 53–69. https://doi.org/10.1016/s0020-0255(01)00092-5

Braun, G. J. (1999). Image lightness rescaling using sigmoidal contrast enhancement functions. Journal of Electronic Imaging, 8(4), 380. https://doi.org/10.1117/1.482706

Benuwa, B. B., Zhan, Y. Z., Ghansah, B., Wornyo, D. K., & Banaseka Kataka, F. (2016). A Review of Deep Machine Learning. *International Journal of Engineering Research in Africa*, *24*, 124–136. https://doi.org/10.4028/www.scientific.net/jera.24.124

Bengio, Y., Courville, A., & Vincent, P. (2012). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1798-1828. https://doi.org/10.1109/TPAMI.2013.50

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, *2*(1), 1–127. https://doi.org/10.1561/2200000006

Ballas, N., Yao, L., Pal, C., & Courville, A. (2016, March 1). *Delving Deeper into Convolutional Networks for Learning Video Representations*. ArXiv.org. https://doi.org/10.48550/arXiv.1511.06432

Bergkvist, A., Rusnakova, V., Sindelka, R., Garda, J. M. A., Sjögreen, B., Lindh, D., Forootan, A., & Kubista, M. (2010). Gene expression profiling – Clusters of possibilities. Methods, 50(4), 323–335. https://doi.org/10.1016/j.ymeth.2010.01.009

Basha, S. H. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2019). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2019.10.008

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Breiman, L. (1984). Classification and regression trees. CRC press.

Brodtkorb, A. R., Hagen, T. R., & Sætra, M. L. (2013). Graphics processing unit (GPU) programming strategies and trends in GPU computing. *Journal of Parallel and Distributed Computing*, *73*(1), 4–13. https://doi.org/10.1016/j.jpdc.2012.04.003

B Arnold, T. (2017). KerasR: R Interface to the Keras Deep Learning Library. *The Journal of Open Source Software*, *2*(14), 296. https://doi.org/10.21105/joss.00296

Boixader, D., & Jacas, J. (1998). Extensionality based approximate reasoning. *International Journal of Approximate Reasoning*, *19*(3-4), 221–230. https://doi.org/10.1016/s0888-613x(98)00018-8

Cheng, G., Zhou, P., & Han, J. (2016). Learning Rotation-Invariant Convolutional Neural Networks for Object Detection in VHR Optical Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, *54*(12), 7405–7415. https://doi.org/10.1109/TGRS.2016.2601622

CIFAR-10 数据集可视化详细讲解（附代码）｜航行学园. (n.d.). Www.voycn.com. Retrieved September 25, 2023, from http://www.voycn.com/article/cifar-10-shujujikeshihuaxiangxijiangjiefudaima

Couprie, C., Farabet, C., Najman, L., & Lecun, Y. (n.d.). *Indoor Semantic Segmentation using depth information*. Retrieved September 13, 2023, from https://arxiv.org/pdf/1301.3572

Chen, Y., Lin, Z., Zhao, X., Wang, G., & Gu, Y. (2014). Deep Learning-Based Classification of Hyperspectral Data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *7*(6), 2094–2107. https://doi.org/10.1109/jstars.2014.2329330

Choe, J.-I., & Shim, H. (2019). Attention-Based Dropout Layer for Weakly Supervised Object Localization. *ArXiv (Cornell University)*. https://doi.org/10.1109/cvpr.2019.00232

Cooil, B., Lerzan Aksoy, & Keiningham, T. L. (2007). Approaches to Customer Segmentation. ResearchGate; Taylor & Francis (Routledge). https://www.researchgate.net/publication/230557972_Approaches_to_Customer_Segmentation

Cho, K., Bart van Merrienboer, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.3115/v1/d14-1179

Chollet, F. (2018). Keras: The Python Deep Learning library. *Astrophysics Source Code Library*.

Cheng, K., Tahir, R., Eric, L. K., & Li, M. (2020). An analysis of generative adversarial networks and variants for image synthesis on MNIST dataset. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-019-08600-2

Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: a taxonomy. *IEEE Software*, *7*(1), 13–17. https://doi.org/10.1109/52.43044

Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., & Ma, Y. (2015). PCANet: A Simple Deep Learning Baseline for Image Classification? *IEEE Transactions on Image Processing*, *24*(12), 5017–5032. https://doi.org/10.1109/tip.2015.2475625

*cifar100 | TensorFlow Datasets*. (n.d.). TensorFlow. Retrieved May 6, 2023, from https://www.tensorflow.org/datasets/catalog/cifar100

Chaoyang, L., Fang, L., & Yinxiang, X. (2003, September 1). Face recognition using self-organizing feature maps and support vector machines. IEEE Xplore. https://doi.org/10.1109/ICCIMA.2003.1238097

Chiu, W.-Y., Yen, G. G., & Juan, T.-K. (2016). Minimum Manhattan Distance Approach to Multiple Criteria Decision Making in Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, *20*(6), 972–985. https://doi.org/10.1109/tevc.2016.2564158

Cao, Z., Li, X., Feng, Y., Chen, S., Xia, C., & Zhao, L. (2021). ContrastNet: Unsupervised feature learning by autoencoder and prototypical contrastive learning for hyperspectral imagery classification. Neurocomputing, 460, 71–83. https://doi.org/10.1016/j.neucom.2021.07.015

*cifar10 | TensorFlow Datasets*. (n.d.). TensorFlow. Retrieved December 29, 2022, from https://www.tensorflow.org/datasets/catalog/cifar10

C. Brian Atkins, Bouman, C. A., & Allebach, J. P. (2002). Optimal image scaling using pixel classification. CiteSeer X (the Pennsylvania State University). https://doi.org/10.1109/icip.2001.958257

Chen, X., Ding, M., Wang, X., Xin, Y., Mo, S., Wang, Y., Han, S., Luo, P., Zeng, G., & Wang, J. (2022). Context Autoencoder for Self-Supervised Representation Learning. Openreview.net. https://openreview.net/forum?id=Gb2Rndy5595

Computer vision: algorithms and applications. (2011). Choice Reviews Online, 48(09), 48–514048–5140. https://doi.org/10.5860/choice.48-5140

Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., & Ma, Y. (2015). PCANet: A Simple Deep Learning Baseline for Image Classification? IEEE Transactions on Image Processing, 24(12), 5017–5032. https://doi.org/10.1109/tip.2015.2475625

Coates, A., & Ng, A. Y. (2012). Learning Feature Representations with K-Means. Lecture Notes in Computer Science, 561–580. https://doi.org/10.1007/978-3-642-35289-8_30

Diniz, W. J. S., & Canduri, F. (2017). REVIEW-ARTICLE Bioinformatics: an overview and its applications. *Genetics and Molecular Research*, *16*(1). https://doi.org/10.4238/gmr16019645

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019* (pp. 4171–4186).

Dugger, S. A., Platt, A., & Goldstein, D. B. (2017). Drug development in the era of precision medicine. *Nature Reviews Drug Discovery*, *17*(3), 183–196. https://doi.org/10.1038/nrd.2017.226

Dorafshan, S., Thomas, R. J., & Maguire, M. (2018). Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Construction and Building Materials*, *186*, 1031–1045. https://doi.org/10.1016/j.conbuildmat.2018.08.011

Digitalcommons@uri, D., & Yuan, L. (n.d.). Implementation of Self-Organizing Maps with Python Implementation of Self-Organizing Maps with Python. Retrieved September 21, 2023, from https://digitalcommons.uri.edu/cgi/viewcontent.cgi?article=2244&context=theses

Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2008). Image retrieval. *ACM Computing Surveys*, *40*(2), 1–60. https://doi.org/10.1145/1348246.1348248

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2009.5206848

Divam. (2019, March 8). *An Overview of Deep Learning Based Clustering Techniques*. Divam Gupta. https://divamgupta.com/unsupervised-learning/2019/03/08/an-overview-of-deep-learning-based-clustering-techniques.html

Deravi, F. (2007). Editorial: IET Image Processing. IET Image Processing, 1(1), 1. https://doi.org/10.1049/iet-ipr:20079006

Dolgikh, S. (2021). Analysis and Augmentation of Small Datasets with Unsupervised Machine Learnin. Europe PMC. https://europepmc.org/article/PPR/PPR316421

Detecting and Combating Deep Fakes. (2021). The Journal of Intelligence, Conflict, and Warfare, 3(3), 83–87. https://doi.org/10.21810/jicw.v3i3.2752

Danuser, G. (2011). Computer Vision in Cell Biology. *Cell*, *147*(5), 973–978. https://doi.org/10.1016/j.cell.2011.11.001

Djokic-Petrovic, M., Pritchard, D., Ivanovic, M., & Cvjetkovic, V. (2016). IMI Python: Upgraded CS Circles web-based Python course. Computer Applications in Engineering Education, 24(3), 464–480. https://doi.org/10.1002/cae.21724

Designing TensorFlow Modeling Code For TFX. (n.d.). TensorFlow. Retrieved September 26, 2023, from https://www.tensorflow.org/tfx/guide/train

Ephrat, A., Mosseri, I., Lang, O., Dekel, T., Wilson, K., Hassidim, A., Freeman, W. T., & Rubinstein, M. (2018). Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation. *ACM Transactions on Graphics*, *37*(4), 1–11. https://doi.org/10.1145/3197517.3201357

Ekanayake, N., & Ranjith Liyanapathirana. (1994). On the exact formula for the minimum squared Euclidean distance of CPFSK. *IEEE Transactions on Communications*, *42*(11), 2917–2918. https://doi.org/10.1109/26.328969

Ertuğrul, Ö. F. (2018). A novel type of activation function in artificial neural networks: Trained activation function. *Neural Networks*, *99*, 148–157. https://doi.org/10.1016/j.neunet.2018.01.007

Ephrat, A., Mosseri, I., Lang, O., Dekel, T., Wilson, K., Hassidim, A., Freeman, W. T., & Rubinstein, M. (2018). Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation. *ACM Transactions on Graphics*, *37*(4), 1–11. https://doi.org/10.1145/3197517.3201357

Emily Edward.(2023).Machine Learning for Medical Diagnostics: A Review in Bioinformatics.

Eisen, M. B., Spellman, P. T., Brown, P. O., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Sciences, 95(25), 14863–14868. https://doi.org/10.1073/pnas.95.25.14863

Eckardt, B. V., & Haugeland, J. (1988). Artificial Intelligence: The Very Idea. *The Philosophical Review*, *97*(2), 286. https://doi.org/10.2307/2185277

Edge Detection Using OpenCV | LearnOpenCV #. (2021, June 10). LearnOpenCV. https://learnopencv.com/edge-detection-using-opencv/

Feng, C.-C., Wang, Y.-C., & Chen, C.-Y. (2013). Combining Geo-SOM and Hierarchical Clustering to Explore Geospatial Data. *Transactions in GIS*, *18*(1), 125–146. https://doi.org/10.1111/tgis.12025

Feng, H.-Y., Chen, H.-W., & Hou, J. (2021). *SR-ScatNet Algorithm for On-device ECG Time Series Anomaly Detection*. https://doi.org/10.1109/southeastcon45413.2021.9401872

Ferles, C., Papanikolaou, Y., Savaidis, S. P., & Mitilineos, S. A. (2021). Deep Self-Organizing Map of Convolutional Layers for Clustering and Visualizing Image Data. Machine Learning and Knowledge Extraction, 3(4), 879–899. https://doi.org/10.3390/make304004

Ernesto, & Hashmi, M. (2021, April 28). What is Data Preprocessing in Machine Learning? | Data Science Process. Dr. Ernesto Lee. https://ernesto.net/data-preprocessing-in-machine-learning/

Forest, F., Mustapha Lebbah, Hanene Azzag, & Lacaille, J. (2021). Deep embedded self-organizing maps for joint representation learning and topology-preserving clustering. Neural Computing and Applications, 33(24), 17439–17469. https://doi.org/10.1007/s00521-021-06331-w

Gerry. (n.d.). *BIRDS 400 - SPECIES IMAGE CLASSIFICATION*. Www.kaggle.com. https://www.kaggle.com/datasets/gpiosenka/100-bird-species

Gilewski, J. (2019, November 9). Modular image processing pipeline using OpenCV and Python generators. DeepVisionGuru. https://medium.com/deepvisionguru/modular-image-processing-pipeline-using-opencv-and-python-generators-9edca3ccb696

Gunes Kayacik, H., Nur Zincir-Heywood, A., & Heywood, M. I. (2007). A hierarchical SOM-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, *20*(4), 439–451. https://doi.org/10.1016/j.engappai.2006.09.005

Guan, D. (2020, July 27). *Classical Architectures in CNN*. Deep Learning. https://guandi1995.github.io/Classical-CNN-architecture/

Goldani, M. H., Momtazi, S., & Safabakhsh, R. (2021). Detecting fake news with capsule neural networks. *Applied Soft Computing*, *101*, 106991. https://doi.org/10.1016/j.asoc.2020.106991

Google. (2019). TensorFlow. TensorFlow; Google. https://www.tensorflow.org/

Gorishniy, Y., Rubachev, I., & Babenko, A. (2023, July 26). *On Embeddings for Numerical Features in Tabular Deep Learning*. ArXiv.org. https://doi.org/10.48550/arXiv.2203.05556

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Giraudel, J. L., & Lek, S. (2001). A comparison of self-organizing map algorithm and some conventional statistical methods for ecological community ordination. *Ecological Modelling*, *146*(1-3), 329–339. https://doi.org/10.1016/s0304-3800(01)00324-6

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 1157-1182.

Grajciarová, L., Mareš, J., Dvořák, P., & Procházka, A. (n.d.). *BIOMEDICAL IMAGE ANALYSIS USING SELF-ORGANIZING MAPS*. Retrieved September 19, 2023, from http://dsp.vscht.cz/konference_matlab/MATLAB12/full_paper/028_Grajciarova.pdf

Gower, J. C. (1985). Properties of Euclidean and non-Euclidean distance matrices. Linear Algebra and Its Applications, 67, 81–97. https://doi.org/10.1016/0024-3795(85)90187-9

Gao, Y., Kang, X., & Chen, Y. (2020). A robust video zero-watermarking based on deep convolutional neural network and self-organizing map in polar complex exponential transform domain. Multimedia Tools and Applications, 80(4), 6019–6039. https://doi.org/10.1007/s11042-020-09904-4

Greenberg-Toledo, T., Mazor, R., Haj-Ali, A., & Kvatinsky, S. (2019). Supporting the Momentum Training Algorithm Using a Memristor-Based Synapse. IEEE Transactions on Circuits and Systems I: Regular Papers, 66(4), 1571–1583. https://doi.org/10.1109/tcsi.2018.2888538

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/cvpr.2016.90

Heaton, J. (2017). Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. *Genetic Programming and Evolvable Machines*, *19*(1-2), 305–307. https://doi.org/10.1007/s10710-017-9314-z

Han, J., Kamber, M., & Pei, J. (2012). Data mining: concepts and techniques. Elsevier.

Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text* Classification. ArXiv.org. https://arxiv.org/abs/1801.06146

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770–778).

Huang, M.-H., & Rust, R. T. (2018). Artificial Intelligence in Service. *Journal of Service Research*, *21*(2), 155–172. https://doi.org/10.1177/1094670517752459

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, *18*(7), 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

Helmstaedter, M., Briggman, K. L., Turaga, S. C., Jain, V., Seung, H. S., & Denk, W. (2013). Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, *500*(7461), 168–174. https://doi.org/10.1038/nature12346

Hinton, G. (2009). Deep belief networks. *Scholarpedia*, *4*(5), 5947. https://doi.org/10.4249/scholarpedia.5947

Habibi Aghdam, H., & Jahani Heravi, E. (2017). *Guide to Convolutional Neural Networks*. Springer International Publishing. https://doi.org/10.1007/978-3-319-57550-6

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hankins, R., Peng, Y., & Yin, H. (2018). SOMNet: Unsupervised Feature Learning Networks for Image Classification. Research Explorer (the University of Manchester). https://doi.org/10.1109/ijcnn.2018.8489404

Hankins, R., Peng, Y., & Yin, H. (2018). SOMNet: Unsupervised Feature Learning Networks for Image Classification. *Research Explorer (the University of Manchester)*. https://doi.org/10.1109/ijcnn.2018.8489404

Horie, N., Matsui, T., Moriyama, K., Mutoh, A., & Inuzuka, N. (2019). Multi-objective safe reinforcement learning: the relationship between multi-objective reinforcement learning and safe reinforcement learning. *Artificial Life and Robotics*. https://doi.org/10.1007/s10015-019-00523-3

Han, X., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. ArXiv (Cornell University).

Haykin, S. (1999). *Neural networks : a comprehensive foundation*. Pearson Education.

Han, K., Wen, H., Shi, J., Lu, K.-H., Zhang, Y., Fu, D., & Liu, Z. (2019). Variational autoencoder: An unsupervised model for encoding and decoding fMRI activity in visual cortex. *NeuroImage*, *198*, 125–136. https://doi.org/10.1016/j.neuroimage.2019.05.039

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Han, X., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. ArXiv (Cornell University).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. Retrieved from https://arxiv.org/abs/1503.02531

Hankins, R., Peng, Y., & Yin, H. (2018). SOMNet: Unsupervised Feature Learning Networks for Image Classification. Research Explorer (the University of Manchester). https://doi.org/10.1109/ijcnn.2018.8489404

Handl, J., Knowles, J., & Kell, D. B. (2005). Computational cluster validation in post-genomic data analysis. Bioinformatics, 21(15), 3201–3212. https://doi.org/10.1093/bioinformatics/bti517

Hemingway, H., Asselbergs, F. W., Danesh, J., Dobson, R., Maniadakis, N., Maggioni, A., van Thiel, G. J. M., Cronin, M., Brobert, G., Vardas, P., Anker, S. D., Grobbee, D. E., & Denaxas, S. (2017). Big data from electronic health records for early and late translational cardiovascular research: challenges and potential. *European Heart Journal*, *39*(16), 1481–1495. https://doi.org/10.1093/eurheartj/ehx487

Iriananda, S. W., Muslim, M. A., & Dachlan, H. S. (2018). Measure the Similarity of Complaint Document Using Cosine Similarity Based on Class-Based Indexing. International Journal of Computer Applications Technology and Research, 7(8), 292–296. https://doi.org/10.7753/ijcatr0708.1001

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ArXiv.org. https://arxiv.org/abs/1502.03167

Image Resizing in Saliency Histogram Domain. (2017). Sensors and Materials, 1483. https://doi.org/10.18494/sam.2017.1657

Jason Brownlee. (2019, July 5). *A Gentle Introduction to Object Recognition With Deep Learning*. Machine Learning Mastery. https://machinelearningmastery.com/object-recognition-with-deep-learning/

Jokhio, F. (2019). Image Classification using AlexNet with SVM Classifier and Transfer Learning. Www.academia.edu. https://www.academia.edu/65400683/Image_Classification_using_AlexNet_with_SVM_Classifier_and_Transfer_Learning

Jiménez, J., Škalič, M., Martínez-Rosell, G., & De Fabritiis, G. (2018). KDEEP: Protein–Ligand Absolute Binding Affinity Prediction via 3D-Convolutional Neural Networks. *Journal of Chemical Information and Modeling*, *58*(2), 287–296. https://doi.org/10.1021/acs.jcim.7b00650

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe. *Proceedings of the ACM International Conference on Multimedia - MM '14*. https://doi.org/10.1145/2647868.2654889

Jaakko Hollmén, Volker Tresp, & Simula, O. (1999). A self-organizing map for clustering probabilistic models. https://doi.org/10.1049/cp:19991234

Joni-Kristian Kämäräinen. (2003). *Feature extraction using Gabor filters*. Lappeenranta Lappeenrannan Teknillinen Yliopisto.

Jäkel, F., & Schreiber, C. (2013). Introspection in Problem Solving. *The Journal of Problem Solving*, *6*(1). https://doi.org/10.7771/1932-6246.1131

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), 651 .666–https://doi.org/10.1016/j.patrec.2009.09.011

Ji, X., Henriques, J. F., & Vedaldi, A. (2018). Invariant Information Distillation for Unsupervised Image Segmentation and Clustering. ArXiv (Cornell University).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Kulak, T., Fillion, A., & Blayo, F. (2022, May 3). A unified view on Self-Organizing Maps (SOMs) and Stochastic Neighbor Embedding (SNE). ArXiv.org. https://doi.org/10.48550/arXiv.2205.01492

Khan, S., & Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265. https://doi.org/10.1016/j.ymssp.2017.11.024

Karel, A. A. (2021, March 16). Ensemble Learning. Nerd for Tech. https://medium.com/nerd-for-tech/ensemble-learning-f93819e0b196

Kaski, S., Honkela, T., Lagus, K., & Kohonen, T. (1998). WEBSOM – Self-organizing maps of document collections. Neurocomputing, 21(1-3), 101–117. https://doi.org/10.1016/s0925-2312(98)00039-3

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480. https://doi.org/10.1109/5.58325

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Networks*, *37*, 52–65. https://doi.org/10.1016/j.neunet.2012.09.018

Kolasa, M., Długosz, R., Pedrycz, W., & Szulc, M. (2012). A programmable triangular neighborhood function for a Kohonen self-organizing map implemented on chip. *Neural Networks*, *25*, 146–160. https://doi.org/10.1016/j.neunet.2011.09.002

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480. https://doi.org/10.1109/5.58325

Krig, S. (2014). Image Pre-Processing. Apress EBooks, 39–83. https://doi.org/10.1007/978-1-4302-5930-5_2

Kohonen, T., Oja, E., Simula, O., Visa, A., & Kangas, J. (1996). Engineering applications of the self-organizing map. *Proceedings of the IEEE*, *84*(10), 1358–1384. https://doi.org/10.1109/5.537105

Kamran Ghasedi Dizaji, Amirhossein Herandi, Deng, C., Cai, W., & Huang, H. (2017). Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. *International Conference on Computer Vision*. https://doi.org/10.1109/iccv.2017.612

Kulikowski, C. A. (2019). Beginnings of Artificial Intelligence in Medicine (AIM): Computational Artifice Assisting Scientific Inquiry and Clinical Art – with Reflections on Present AIM Challenges. *Yearbook of Medical Informatics*, *28*(01), 249–256. https://doi.org/10.1055/s-0039-1677895

Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464–1480. https://doi.org/10.1109/5.58325

Karaev, A., Lenny Koh, S. C., & Szamosi, L. T. (2007). The cluster approach and SME competitiveness: a review. Journal of Manufacturing Technology Management, 18(7), 818 .835–https://doi.org/10.1108/17410380710817273

Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., & Saarela, A. (2000). Self organization of a massive document collection. IEEE Transactions on Neural Networks, 11(3), 574 .585–https://doi.org/10.1109/72.846729

Krut Patel. (2019, September 7). MNIST Handwritten Digits Classification using a Convolutional Neural Network (CNN). Medium; Towards Data Science. https://towardsdatascience.com/mnist-handwritten-digits-classification-using-a-convolutional-neural-network-cnn-af5fafbc35e9

LEGENDRE, P., & FORTIN, M.-J. (2010). Comparison of the Mantel test and alternative approaches for detecting complex multivariate relationships in the spatial analysis of genetic data. Molecular Ecology Resources, 10(5), 831–844. https://doi.org/10.1111/j.1755-0998.2010.02866.x

Lu, Y., & Young, S. (2020). A survey of public datasets for computer vision tasks in precision agriculture. Computers and Electronics in Agriculture, 178, 105760. https://doi.org/10.1016/j.compag.2020.105760

Li, H., Liu, H., Ji, X., Li, G., & Shi, L. (2017). CIFAR10-DVS: An Event-Stream Dataset for Object Classification. Frontiers in Neuroscience, 11. https://doi.org/10.3389/fnins.2017.00309

Lin, S.-K. (2011). Social Sciences and Sustainability. Social Sciences, 1(1), 1–1. https://doi.org/10.3390/socsci1010001

Lawrence, S., Giles, C. L., Ah Chung Tsoi, & Back, A. D. (1997). Face recognition: a convolutional neural-network approach. IEEE Transactions on Neural Networks, 8(1), 98–113. https://doi.org/10.1109/72.554195

Lee, J. A., & Verleysen, M. (2002). Self-organizing maps with recursive neighborhood adaptation. *Neural Networks*, *15*(8-9), 993–1003. https://doi.org/10.1016/s0893-6080(02)00073-4

López, M., Valero, S., Senabre, C., Aparicio, J., & Gabaldon, A. (2012). Application of SOM neural networks to short-term load forecasting: The Spanish electricity market case study. *Electric Power Systems Research*, *91*, 18–27. https://doi.org/10.1016/j.epsr.2012.04.009

Laaksonen, J., Koskela, M., & Oja, E. (2002). PicSOM-self-organizing image retrieval with MPEG-7 content descriptors. *IEEE Transactions on Neural Networks*, *13*(4), 841–853. https://doi.org/10.1109/tnn.2002.1021885

Li, X., Zhang, Y., Cheng, H., Li, M., & Yin, B. (2022). Student achievement prediction using deep neural network from multi-source campus data. Complex & Intelligent Systems. https://doi.org/10.1007/s40747-022-00731-8

Layek, & Mukhopadhyay. (1977). On Cluster Sets And Essential Cluster Sets. *Real Analysis Exchange*, *3*(1), 25. https://doi.org/10.2307/44151122

Lei, B., Huang, S., Li, R., Bian, C., Li, H., Chou, Y.-H., & Cheng, J.-Z. (2018). Segmentation of breast anatomy for automated whole breast ultrasound images with boundary regularized convolutional encoder–decoder network. *Neurocomputing*, *321*, 178–186. https://doi.org/10.1016/j.neucom.2018.09.043

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liu, Q., & Wu, Y. (2012). Supervised Learning. *Encyclopedia of the Sciences of Learning*, 3243–3245. https://doi.org/10.1007/978-1-4419-1428-6_451

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liu, C., & Zhang, L. (2023). A Novel Denoising Algorithm Based on Wavelet and Non-Local Moment Mean Filtering. Electronics, 12(6), 1461. https://doi.org/10.3390/electronics12061461

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Liu, Y., Zhang, J., Xiong, H., Zhou, L., He, Z., Wu, H., Wang, H., & Zong, C. (2020). Synchronous Speech Recognition and Speech-to-Text Translation with Interactive Decoding. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 8417–8424. https://doi.org/10.1609/aaai.v34i05.6360

Lee, K.-S., Turner, N. J., Macrina, T., Wu, J., Lu, R., & H. Sebastian Seung. (2019). *Convolutional nets for reconstructing neural circuits from brain images acquired by serial section electron microscopy*. *55*, 188–198. https://doi.org/10.1016/j.conb.2019.04.001

Liberti, L., Lavor, C., Maculan, N., & Mucherino, A. (2012, May 2). *Euclidean distance geometry and applications*. ArXiv.org. https://doi.org/10.48550/arXiv.1205.0349

Li, A., Chen, D., Wu, Z., Sun, G., & Lin, K. (2018). Self-supervised sparse coding scheme for image classification based on low rank representation. *PLOS ONE*, *13*(6), e0199141–e0199141. https://doi.org/10.1371/journal.pone.0199141

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

*LeCun, Y., et al. (2015) Deep Learning. Nature, 521, 436-444. - References - Scientific Research Publishing*. (n.d.). Www.scirp.org. https://www.scirp.org/reference/referencespapers.aspx?referenceid=2704915https://doi.org/10.1038/nature1453

Le Roux, N., & Bengio, Y. (2008). Representational Power of Restricted Boltzmann Machines and Deep Belief Networks. *Neural Computation*, *20*(6), 1631–1649. https://doi.org/10.1162/neco.2008.04-07-510

M. Ben-Ari and F. Mondada.(2018). *(PDF) Robots and Their Applications*. (n.d.). ResearchGate. https://www.researchgate.net/publication/320674637_Robots_and_Their_Applications

Madhuri, G. S., & Rani, M. U. (2018). Anomaly Detection Techniques. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3167172

Mohanty, S. P., Choppali, U., & Kougianos, E. (2016). Everything you wanted to know about smart cities: The Internet of things is the backbone. *IEEE Consumer Electronics Magazine*, *5*(3), 60–70. https://doi.org/10.1109/mce.2016.2556879

Manduchi, L., Hüser, M., Vogt, J., Rätsch, G., & Fortuin, V. (2020, June 9). DPSOM: Deep Probabilistic Clustering with Self-Organizing Maps. ArXiv.org. https://doi.org/10.48550/arXiv.1910.01590

Miller, M. (2015). *The Internet of Things*. Que Publishing.

Merchant, A., Rahimtoroghi, E., Pavlick, E., & Tenney, I. (n.d.). *What Happens To BERT Embeddings During Fine-tuning?* https://arxiv.org/pdf/2004.14448.pdf

Mert Copur, Buse MelisOzyildirim, & Turgay Ibrikci. (2018). *Image Classification of Aerial Images Using CNN-SVM*. https://doi.org/10.1109/asyu.2018.8554008

Miura, J., Kanda, T., & Shirai, Y. (2000, October 1). *An active vision system for real-time traffic sign recognition*. IEEE Xplore. https://doi.org/10.1109/ITSC.2000.881017

Mathieu, M., Couprie, C., & Lecun, Y. (n.d.). *DEEP MULTI-SCALE VIDEO PREDICTION BEYOND MEAN SQUARE ERROR*. https://arxiv.org/pdf/1511.05440v3

Merrill, N., & Eskandarian, A. (2020). Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning. IEEE Access, 1–1. https://doi.org/10.1109/access.2020.2997327

Natita, W., Wiboonsak, W., & Dusadee, S. (2016). Appropriate Learning Rate and Neighborhood Function of Self-organizing Map (SOM) for Specific Humidity Pattern Classification over Southern Thailand. *International Journal of Modeling and Optimization*, *6*(1), 61–65. https://doi.org/10.7763/ijmo.2016.v6.504

Miljković, D. (2017, May 1). *Brief review of self-organizing maps*. IEEE Xplore. https://doi.org/10.23919/MIPRO.2017.7973581

Mao, Y., & Yin, Z. (2016). A Hierarchical Convolutional Neural Network for Mitosis Detection in Phase-Contrast Microscopy Images. *Lecture Notes in Computer Science*, 685–692. https://doi.org/10.1007/978-3-319-46723-8_79

Mei, S., Ji, J., Geng, Y., Zhang, Z., Li, X., & Du, Q. (2019). Unsupervised Spatial–Spectral Feature Learning by 3D Convolutional Autoencoder for Hyperspectral Classification. 57(9), 6808–6820. https://doi.org/10.1109/tgrs.2019.2908756

Manduchi, L., Hüser, M., Vogt, J., Rätsch, G., & Fortuin, V. (2020, June 9). DPSOM: Deep Probabilistic Clustering with Self-Organizing Maps. ArXiv.org. https://doi.org/10.48550/arXiv.1910.01590

Malkauthekar, M. D. (2013). Analysis of euclidean distance and manhattan distance measure in face recognition. Third International Conference on Computational Intelligence and Information Technology (CIIT 2013). https://doi.org/10.1049/cp.2013.2636

Mohanty, R., Kumar Mallik, B., & Singh Solanki, S. (2020). Recognition of Bird Species Based on Spike Model Using Bird Dataset. Data in Brief, 105301. https://doi.org/10.1016/j.dib.2020.105301

Madhulatha, T. S. (2012). An Overview on Clustering Methods. Arxiv.org. https://arxiv.org/abs/1205.1117

Ngo, G. C., & Macabebe, E. Q. B. (2016, November 1). Image segmentation using K-means color quantization and density-based spatial clustering of applications with noise (DBSCAN) for hotspot detection in photovoltaic modules. IEEE Xplore. https://doi.org/10.1109/TENCON.2016.7848290

Nguyen-Phuoc, T., Ding, Y., Theis, L., Richardt, C., & Yang, Y.-L. (2019). *HoloGAN: Unsupervised Learning of 3D Representations From Natural Images*. https://doi.org/10.1109/iccvw.2019.00255

Normalization Formula | Step By Step Guide with Calculation Examples. (2019, May 15). WallStreetMojo. https://www.wallstreetmojo.com/normalization-formula/

N., Sam M.S.(2013).*What is BIOLOGICAL INTELLIGENCE? definition of BIOLOGICAL INTELLIGENCE (Psychology Dictionary)*. (2013, April 7). https://psychologydictionary.org/biological-intelligence/

Norgeot, B., Glicksberg, B. S., Trupin, L., Lituiev, D., Gianfrancesco, M., Oskotsky, B., Schmajuk, G., Yazdany, J., & Butte, A. J. (2019). Assessment of a Deep Learning Model Based on Electronic Health Record Data to Forecast Clinical Outcomes in Patients With Rheumatoid Arthritis. *JAMA Network Open*, *2*(3), e190606. https://doi.org/10.1001/jamanetworkopen.2019.0606

Natita, W., Wiboonsak, W., & Dusadee, S. (2016). Appropriate Learning Rate and Neighborhood Function of Self-organizing Map (SOM) for Specific Humidity Pattern Classification over Southern Thailand. *International Journal of Modeling and Optimization*, *6*(1), 61–65. https://doi.org/10.7763/ijmo.2016.v6.504

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *ArXiv:1511.08458 [Cs]*. https://arxiv.org/abs/1511.08458

Odena, A. (2019). Open Questions about Generative Adversarial Networks. *Distill*, *4*(4). https://doi.org/10.23915/distill.00018

O'shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*. https://arxiv.org/pdf/1511.08458

Oak, R. (2016). A Study of Digital Image Segmentation Techniques. International Journal of Engineering and Computer Science. https://doi.org/10.18535/ijecs/v5i12.76

Pak, M., & Kim, S. (2017, August 1). *A review of deep learning in image recognition*. IEEE Xplore. https://doi.org/10.1109/CAIPT.2017.8320684

Patro, S. Gopal. K., & sahu, K. K. (2015). Normalization: A Preprocessing Stage. IARJSET, 20–22. https://doi.org/10.17148/iarjset.2015.2305

Preprocess images using OpenCV for pytesseract OCR. (n.d.). Stack Overflow. Retrieved September 26, 2023, from https://stackoverflow.com/questions/63845174/preprocess-images-using-opencv-for-pytesseract-ocr

Perspectives and Future Outlook of Deep Learning AI. (2017). *International Journal of Modern Trends in Engineering & Research*, *4*(11), 86–94. https://doi.org/10.21884/ijmter.2017.4349.pqhil

Prasad, V. (2015). VOICE RECOGNITION SYSTEM: SPEECH-TO-TEXT. Journal of Applied

and Fundamental Sciences, 1(2), 191.

Park, J., Lee, M., Hyung Jin Chang, Lee, K., & Choi, J.-Y. (2019). Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning. International Conference on Computer Vision. https://doi.org/10.1109/iccv.2019.00662

Pu, Y., Min, M., Gan, Z., & Carin, L. (2018). Adaptive Feature Abstraction for Translating Video to Text. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). https://doi.org/10.1609/aaai.v32i1.12245

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering,* *22*(10), 1345-1359. https://doi.org/10.1109/TKDE.2009.191

Pandian, J. A., Kanchanadevi, K., Kumar, V. D., Jasińska, E., Goňo, R., Leonowicz, Z., & Jasiński, M. (2022). A Five Convolutional Layer Deep Convolutional Neural Network for Plant Leaf Disease Detection. Electronics, 11(8), 1266. https://doi.org/10.3390/electronics11081266

Powers, D. M. (2011). Evaluation: From precision, recall, and F-measure to ROC, informedness, markedness, and correlation.Journal of Machine Learning Technologies, 2(1), 37-63.

Pei, S.-C., & Lin, C.-N. (1995). Image normalization for pattern recognition. Image and Vision Computing, 13(10), 711–723. https://doi.org/10.1016/0262-8856(95)98753-g.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations*. ArXiv.org. https://arxiv.org/abs/1802.05365

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Luca Antiga, Alban Desmaison, Kopf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Sasank Chilamkurthy, Steiner, B., Fang, L., & Bai, J. (2019).

PyTorch: An Imperative Style, High-Performance Deep Learning Library. *ArXiv (Cornell University)*, *32*, 8026–8037.

Principe, J. C., Risto Miikkulainen, & Springerlink (Online Service. (2009). Advances in Self-Organizing Maps: 7th International Workshop, WSOM 2009, St. Augustine, Florida, June 8-10, 2009. Proceedings. Springer Berlin Heidelberg.

Papageorgiou, E. I., Stylios, C., & Groumpos, P. P. (2006). Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links. *International Journal of Human-Computer Studies*, *64*(8), 727–743. https://doi.org/10.1016/j.ijhcs.2006.02.009

Polat, O., & Dokur, Z. (2016). Protein Fold Recognition Using Self-Organizing Map Neural Network. *Current Bioinformatics*, *11*(4), 451–458. https://doi.org/10.2174/1574893611666160617091142

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345–1359.

Pulkit Sharma. (n.d.). Analytics Vidhya. Retrieved September 26, 2023, from https://www.analyticsvidhya.com/blog/author/pulkits/

pinsky, eugene. (2018). Mathematical Foundation for Ensemble Machine Learning and Ensemble Portfolio Analysis. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3243974

Qin, C., Schlemper, J., Caballero, J., Price, A. N., Hajnal, J. V., & Rueckert, D. (2019). Convolutional Recurrent Neural Networks for Dynamic MR Image Reconstruction. *IEEE Transactions on Medical Imaging*, *38*(1), 280–290. https://doi.org/10.1109/tmi.2018.2863670

Qu, R., Xu, G., Ding, C., Jia, W., & Sun, M. (2020). Standard Plane Identification in Fetal Brain Ultrasound Scans Using a Differential Convolutional Neural Network. *IEEE Access*, *8*, 83821–83830. https://doi.org/10.1109/access.2020.2991845

Qin, L., Zheng, Q., Jiang, S., Huang, Q., & Gao, W. (2008). Unsupervised texture classification: Automatically discover and classify texture patterns. Image and Vision Computing, 26(5), 647–656. https://doi.org/10.1016/j.imavis.2007.08.003

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). FitNets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*. Retrieved from https://arxiv.org/abs/1412.6550

Ruder, S., Ghaffari, P., & Breslin, J. G. (2016). INSIGHT-1 at SemEval-2016 Task 5: Deep Learning for Multilingual Aspect-based Sentiment Analysis. *North American Chapter of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/s16-1053

Rampasek, L., & Goldenberg, A. (2016). TensorFlow: Biology's Gateway to Deep Learning? Cell Systems, 2(1), 12–14. https://doi.org/10.1016/j.cels.2016.01.009

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pretraining. *OpenAI Research*.

Rebai, I., & BenAyed, Y. (2015). Text-to-speech synthesis system with Arabic diacritic recognition system. *Computer Speech & Language*, *34*(1), 43–60. https://doi.org/10.1016/j.csl.2015.04.002

Rawat, W., & Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, *29*(9), 2352–2449. https://doi.org/10.1162/neco_a_00990

Ralbovsky, N. M., & Lednev, I. K. (2020). Towards development of a novel universal medical diagnostic method: Raman spectroscopy and machine learning. *Chemical Society Reviews*, *49*(20), 7428–7453. https://doi.org/10.1039/D0CS01019G

Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, Harsha Kumara, & Yu, X. (2020). Hierarchical Two-Stream Growing Self-Organizing Maps With Transience for Human Activity Recognition. 16(12), 7756–7764. https://doi.org/10.1109/tii.2019.2957454

Riese, F. M., Keller, S., & Hinz, S. (2019). Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data. Remote Sensing, 12(1), 7. https://doi.org/10.3390/rs12010007

Reis, J., & Housley, M. (2022). *Fundamentals of Data Engineering*. "O'Reilly Media, Inc

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*.

*stl10 | TensorFlow Datasets*. (n.d.). TensorFlow. Retrieved May 6, 2023, from https://www.tensorflow.org/datasets/catalog/stl10

Shafiq, M., Ji, L., Liu, A. X., & Wang, J. (2011). Characterizing and modeling internet traffic dynamics of cellular devices. *Measurement and Modeling of Computer Systems*. https://doi.org/10.1145/1993744.1993776

STL-10 dataset. (n.d.). Cs.stanford.edu. https://cs.stanford.edu/~acoates/stl10/

Sakkari, M., & Zaied, M. (2020). A Convolutional Deep Self-Organizing Map Feature extraction for machine learning. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-020-08822-9

Sun, M., Song, Z., Jiang, X., Pan, J., & Pang, Y. (2017). Learning Pooling for Convolutional Neural Network. Neurocomputing, 224, 96–104. https://doi.org/10.1016/j.neucom.2016.10.049

Sekhon, M. (2019, November 24). Image Filters in Python. Medium. https://towardsdatascience.com/image-filters-in-python-26ee938e57d2

Sul, S.-J., & Andrey Tovchigrechko. (2011). Parallelizing BLAST and SOM Algorithms with MapReduce-MPI Library. https://doi.org/10.1109/ipdps.2011.180

Shorten, C. (2019, April 14). Unsupervised Feature Learning. Medium. https://towardsdatascience.com/unsupervised-feature-learning-46a2fe399929

Smutny, P., & Schreiberova, P. (2020). Chatbots for learning: A review of educational chatbots for the Facebook Messenger. Computers & Education, 151(103862), 103862. https://doi.org/10.1016/j.compedu.2020.103862

Schuld, M., & Petruccione, F. (2018). Supervised Learning with Quantum Computers. In *The Open Library*. Springer. https://openlibrary.org/books/OL27764408M/Supervised_Learning_with_Quantum_Computers

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2020). Efficient Processing of Deep Neural Networks. In *Synthesis Lectures on Computer Architecture*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01766-7

Schmidhuber, J. (2015). Deep Learning. *Scholarpedia*, *10*(11), 32832. https://doi.org/10.4249/scholarpedia.32832

Siadati, S. (2018). What is UnSupervised Learning. *Www.academia.edu*. https://www.academia.edu/43389195/What_is_UnSupervised_Learning

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., & Vanhoucke, V. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9. https://doi.org/10.1109/cvpr.2015.7298594

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Salakhutdinov, R., & Hinton, G. (2012). An Efficient Learning Procedure for Deep Boltzmann Machines. *Neural Computation*, *24*(8), 1967–2006. https://doi.org/10.1162/neco_a_00311

Shlens, J. (2014). A Tutorial on Principal Component Analysis. https://arxiv.org/pdf/1404.1100.pdf

Sunil Kumar, A. S., & Mahesh, K. (2023, July 1). *An Investigation of Deep Neural Network based Techniques for Object Detection and Recognition Task in Computer Vision*. IEEE Xplore. https://doi.org/10.1109/ICECAA58104.2023.10212307

Sun, K., Xiao, B., Liu, D., & Wang, J. (2019, June 1). *Deep High-Resolution Representation Learning for Human Pose Estimation*. IEEE Xplore. https://doi.org/10.1109/CVPR.2019.00584

Sinaga, K. P., & Yang, M.-S. (2020). Unsupervised K-Means Clustering Algorithm. IEEE Access, 8, 80716–80727. https://doi.org/10.1109/access.2020.2988796

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic Routing Between Capsules. *ArXiv:1710.09829 [Cs]*. https://arxiv.org/abs/1710.09829

Sarıgül, M., Ozyildirim, B. M., & Avci, M. (2019). Differential convolutional neural network. *Neural Networks*, *116*, 279–287. https://doi.org/10.1016/j.neunet.2019.04.025

Serge Dolgikh. (2021). Analysis and Augmentation of Small Datasets with Unsupervised Machine Learning. https://doi.org/10.1101/2021.04.21.21254796

Saba Jamalian, & Rajaei, H. (2015). Data-Intensive HPC Tasks Scheduling with SDN to Enable HPC-as-a-Service. *Zenodo (CERN European Organization for Nuclear Research)*. https://doi.org/10.1109/cloud.2015.85

Suzuki, K., Roseboom, W., Schwartzman, D. J., & Seth, A. K. (2017). A Deep-Dream Virtual Reality Platform for Studying Altered Perceptual Phenomenology. Scientific Reports, 7(1). https://doi.org/10.1038/s41598-017-16316-2

Sotiris Kotsiantis. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica (Lithuanian Academy of Sciences)*, *31*(3), 249–268.

Siadati, S. (2018). What is UnSupervised Learning. *Www.academia.edu*. https://www.academia.edu/43389195/What_is_UnSupervised_Learning

Sakkari, M., Ejbali, R., & Zaied, M. (2017). Deep SOMs for automated feature extraction and classification from big data streaming. NASA ADS, 10341, 103412L. https://doi.org/10.1117/12.2269082

Toshihiro Takahashi.2018.Image Classification Algorithm Based on Sparse Coding. Journal of Multimedia pp.114 - 122.

https://patentimages.storage.googleapis.com/78/b0/dd/19afb912be2fb1/US10013644.pdf

TAKANASHI, M., TORIKAI, H., & SAITO, T. (2007). An Approach to Collaboration of Growing Self-Organizing Maps and Adaptive Resonance Theory Maps. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E90-A*(9), 2047–2050. https://doi.org/10.1093/ietfec/e90-a.9.2047

Trevino, A. (2016, December 6). *Introduction to K-means Clustering*. Oracle.com. https://blogs.oracle.com/ai-and-datascience/post/introduction-to-k-means-clustering

TensorFlow Datasets. (2010). *mnist | TensorFlow Datasets*. TensorFlow. https://www.tensorflow.org/datasets/catalog/mnist

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 6105–6114).

Tripathy, B. K., S Anveshrithaa, & Shrusti Ghela. (2021). t-Distributed Stochastic Neighbor Embedding (t-SNE). CRC Press EBooks, 127–135. https://doi.org/10.1201/9781003190554-13

Tropp, J., & Baetzgen, A. (2019). Users' Definition of Snapchat Usage. Implications for Marketing on Snapchat. International Journal on Media Management, 21(2), 130–156. https://doi.org/10.1080/14241277.2019.1637343

Teuvo Kohonen. (2012). Self-Organizing Maps. Springer Science & Business Media.

Tian, C., Xu, Y., Li, Z., Zuo, W., Fei, L., & Liu, H. (2020). Attention-guided CNN for image denoising. *Neural Networks*, *124*, 117–129. https://doi.org/10.1016/j.neunet.2019.12.024

Teuvo Kohonen. (2001). Self-organizing maps. New York Springer.

Tan, X., Chen, S., Zhou, Z.-H. ., & Zhang, F. (2005). Recognizing Partially Occluded, Expression Variant Faces From Single Training Image per Person With SOM and Soft$k$-NN Ensemble. IEEE Transactions on Neural Networks, 16(4), 875–886. https://doi.org/10.1109/tnn.2005.849817

T. Kohonen, Self-Organizing Maps, 2nd ed., Springer 1997

T. Kohonen,Unigrafia, Helsinki, Finland.(2014). MATLAB Implementations and Applications of the Self-Organizing Map.

Ulku, I., & Akagündüz, E. (2022). A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D Images. *Applied Artificial Intelligence*, 1–45. https://doi.org/10.1080/08839514.2022.2032924

Van Engelen, J. E., & Hoos, H. H. (2019). A survey on semi-supervised learning. *Machine Learning*. https://doi.org/10.1007/s10994-019-05855-6

Valavanis, I., & Kosmopoulos, D. (2010). Multiclass defect detection and classification in weld radiographic images using geometric and texture features. *Expert Systems with Applications*, *37*(12), 7606–7614. https://doi.org/10.1016/j.eswa.2010.04.082

Vapnik, V. (1995). The nature of statistical learning theory. Springer Science & Business Media.

Vijayakumar, C., Damayanti, G., Pant, R., & Sreedhar, C. M. (2007). Segmentation and grading of brain tumors on apparent diffusion coefficient images using self-organizing maps. *Computerized Medical Imaging and Graphics*, *31*(7), 473–484. https://doi.org/10.1016/j.compmedimag.2007.04.004

Valova, I., Georgiev, G., Gueorguieva, N., & Olson, J. (2013). Initialization Issues in Self-organizing Maps. *Procedia Computer Science*, *20*, 52–57. https://doi.org/10.1016/j.procs.2013.09.238

Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, *11*(3), 586–600. https://doi.org/10.1109/72.846731

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1-3), 37–52. https://doi.org/10.1016/0169-7439(87)80084-9

Wang, L., & Pu, J. (2014). Image Classification Algorithm Based on Sparse Coding. *Journal of Multimedia*, *9*(1). https://doi.org/10.4304/jmm.9.1.114-122

Weng, R., Lu, J., Tan, Y.-P., & Zhou, J. (2016). Learning Cascaded Deep Auto-Encoder Networks for Face Alignment. *IEEE Transactions on Multimedia*, *18*(10), 2066–2078. https://doi.org/10.1109/tmm.2016.2591508

Wang, B., Sun, Y., Xue, B., & Zhang, M. (n.d.). A Hybrid Differential Evolution Approach to Designing Deep Convolutional Neural Networks for Image Classification. Retrieved September 21, 2023, from https://arxiv.org/pdf/1808.06661.pdf

Wikipedia Contributors. (2019, May 4). Python (programming language). Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Python_(programming_language)

Waseem Khan, M. (2014). A Survey: Image Segmentation Techniques. *International Journal of Future Computer and Communication*, 89–93. https://doi.org/10.7763/ijfcc.2014.v3.274

Wen, L., Gao, L., & Li, X. (2019). A New Deep Transfer Learning Based on Sparse Auto-Encoder for Fault Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(1), 136–144. https://doi.org/10.1109/tsmc.2017.2754287

Wouter Van Gansbeke, Vandenhende, S., Georgoulis, S., Proesmans, M., & Luc Van Gool. (2020). Learning To Classify Images Without Labels. *ArXiv (Cornell University)*.

Wickramasinghe, C. S., Amarasinghe, K., & Manic, M. (2017, November 1). Parallalizable deep self-organizing maps for image classification. IEEE Xplore. https://doi.org/10.1109/SSCI.2017.8285443

Witten, I. H., Frank, E., & Hall, M. A. (2011). Data Mining: Practical Machine Learning Tools and Techniques. In Google Books. Elsevier. https://books.google.com.tr/books/about/Data_Mining.html?id=bDtLM8CODsQC&redir_esc=y

Wasserbacher, H., & Spindler, M. (2021). Machine learning for financial forecasting, planning and analysis: recent developments and pitfalls. *Digital Finance*, *4*. https://doi.org/10.1007/s42521-021-00046-2

Wickramasinghe, C. S., Amarasinghe, K., & Manic, M. (2019). Deep Self-Organizing Maps for Unsupervised Image Classification. IEEE Transactions on Industrial Informatics, 15(11), 5837–5845. https://doi.org/10.1109/tii.2019.2906083

Xie, X., & Lam, K.-M. (2008). Face recognition using elastic local reconstruction based on a single face image. *Pattern Recognition*, *41*(1), 406–417. https://doi.org/10.1016/j.patcog.2007.03.020

Xuejun Zhang, X. Z., Xuejun Zhang, J. G., Jiyang Gai, Z. M., Zhili Ma, J. Z., Jinxiong Zhao, H. M., Hongzhong Ma, F. H., & Fucun He, T. J. (2022). Exploring Unsupervised Learning with Clustering and Deep Autoencoder to Detect DDoS Attack. 電腦學刊, 33(4), 029–044. https://doi.org/10.53106/199115992022083304003

Yasaka, K., Akai, H., Abe, O., & Kiryu, S. (2018). Deep Learning with Convolutional Neural Network for Differentiation of Liver Masses at Dynamic Contrast-enhanced CT: A Preliminary Study. *Radiology*, *286*(3), 887–896. https://doi.org/10.1148/radiol.2017170706

Yim, J., Joo, D., Bae, J., & Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization, and transfer learning. In *Proceedings of the IEEE Conference on*

*Computer Vision and Pattern Recognition (CVPR)* (pp. 7130–7138). Retrieved from https://openaccess.thecvf.com/content_cvpr_2017/html/Yim_A_Gift_From_CVPR_2017_paper.html

Yang, Y., Etesami, J., & Kiyavash, N. (2015). *Efficient Neighborhood Selection for Gaussian Graphical Models*. https://arxiv.org/pdf/1509.06449.pdf

Yadav, S. (2020, May 29). A Quick Overview of Contrast Enhancement and Its Variants for Medical Image Processing. Medium. https://medium.com/@sunil7545/a-quick-overview-of-contrast-enhancement-and-its-variants-for-medical-image-processing-fcece3d2298a

Yücesoy, Y. E. Y., & Tümer, M. B. (2015). Hierarchical Reinforcement Learning with Context Detection (HRL-CD). *International Journal of Machine Learning and Computing*, *5*(5), 353–358. https://doi.org/10.7763/ijmlc.2015.v5.533

Yan, L., Zhao, M., Wang, X., Zhang, Y., & Chen, J. (2021). Object Detection in Hyperspectral Images. *IEEE Signal Processing Letters*, *28*, 508–512. https://doi.org/10.1109/LSP.2021.3059204

Yan, K., Li, T., Marques, J. A. L., Gao, J., & Fong, S. J. (2023). A review on multimodal machine learning in medical diagnostics. *Mathematical Biosciences and Engineering: MBE*, *20*(5), 8708–8726. https://doi.org/10.3934/mbe.2023382

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems* (pp. 3320–3328).

Yousefi-Azar, M., Varadharajan, V., Hamey, L., & Tupakula, U. (2017). Autoencoder-based feature learning for cyber security applications. *2017 International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/ijcnn.2017.796634

Yin, Z., Zhao, M., Wang, Y., Yang, J., & Zhang, J. (2017). Recognition of emotions using multimodal physiological signals and an ensemble deep learning model. Computer Methods and Programs in Biomedicine, 140, 93–110. https://doi.org/10.1016/j.cmpb.2016.12.005

Yuda, R. P., Aroef, C., Rustam, Z., & Alatas, H. (2020). Gender Classification Based on Face Recognition using Convolutional Neural Networks (CNNs). *Journal of Physics: Conference Series*, *1490*, 012042. https://doi.org/10.1088/1742-6596/1490/1/012042

Yi Xie, & Shun-Zheng Yu. (2009). Monitoring the Application-Layer DDoS Attacks for Popular Websites. IEEE/ACM Transactions on Networking, 17(1), 15–25. https://doi.org/10.1109/tnet.2008.925628

Zhou, D.-X. (2020). Theory of deep convolutional neural networks: Downsampling. *Neural Networks*. https://doi.org/10.1016/j.neunet.2020.01.018

Zhao, Z., Xu, S., Kang, B. H., Kabir, M. M. J., Liu, Y., & Wasinger, R. (2015). Investigation and improvement of multi-layer perceptron neural networks for credit scoring. *Expert Systems with Applications*, *42*(7), 3508–3516. https://doi.org/10.1016/j.eswa.2014.12.006

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8–36. https://doi.org/10.1109/mgrs.2017.2762307

ZHENG, D., & WANG, Q. (2013). Selection algorithm for K-means initial clustering center. *Journal of Computer Applications*, *32*(8), 2186–2188. https://doi.org/10.3724/sp.j.1087.2012.02186

Zivkovic, Z. (2004, August 1). Improved adaptive Gaussian mixture model for background subtraction. IEEE Xplore. https://doi.org/10.1109/ICPR.2004.1333992

Zhao, W., Wu, C., Yin, K., Young, T. Y., & Ginsberg, M. D. (2006). Pixel-based statistical analysis by a 3D clustering approach: Application to autoradiographic images. Computer Methods and Programs in Biomedicine, 83(1), 18–28. https://doi.org/10.1016/j.cmpb.2006.05.005

Zagoruyko, S., & Komodakis, N. (2016). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*. Retrieved from https://arxiv.org/abs/1612.03928

Zhang, R., Isola, P., & Efros, A. A. (2017). Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction. https://doi.org/10.1109/cvpr.2017.76

Zahrotun, L. (2016). Comparison Jaccard similarity, Cosine Similarity and Combined Both of the Data Clustering With Shared Nearest Neighbor Method. Computer Engineering and Applications Journal, 5(1), 11–18. https://doi.org/10.18495/comengapp.v5i1.160

Zhu, Q., & Zhang, R. (2020, January 10). A Classification Supervised Auto-Encoder Based on Predefined Evenly-Distributed Class Centroids. ArXiv.org. https://doi.org/10.48550/arXiv.1902.00220

Zhu, Q., & Zhang, R. (2019). A Classification Supervised Auto-Encoder Based on Predefined Evenly-Distributed Class Centroids. ArXiv (Cornell University).

Zhao, Z.-Q., & Zhang, X. (2015). Stacked Multilayer Self-Organizing Map for Background Modeling. IEEE Transactions on Image Processing, 24(9), 2841–2850. https://doi.org/10.1109/tip.2015.2427519

Zeyad, D. T. (2005). Human Face Recognition Using GABOR Filter And Different Self Organizing Maps Neural Networks. Al-Khwarizmi Engineering Journal, 1(1), 38–45. https://alkej.uobaghdad.edu.iq/index.php/alkej/article/view/19#:~:text=This%20work%20implements%20the%20face%20recognition%20system%20based

Zhou, Z.-H., Wu, J., & Tang, W. (2002). Ensembling neural networks: Many could be better than all. Artificial Intelligence, 137(1-2), 239–263. https://doi.org/10.1016/s0004-3702(02)00190-x

## CURRICULUM VITAE

Zkeia Abdalla Abdrhman JAZAM, completed her elementary education at Al-Mustagbal Elementary School and her secondary education at Al-Shatti Secondary School. In 2015, she earned a bachelor's degree in Computer Science from Omdurman Islamic University in Sudan. Shortly after graduation, she began her academic career as an Assistant Lecturer at the University of Al-Geneina (GU) in Sudan (http://gu.edu.sd/). In 2018, she obtained her Master’s degree in Information Technology from Al-Neelain University in Sudan. She continued her role as a Lecturer at the University of Al-Geneina until September 2019, when she commenced her Ph.D. studies in the Department of Computer Engineering at Çukurova University, Turkiye. Her doctoral studies were supported by an international scholarship (YÖK scholarship). During her Ph.D., she focused on developing and implementing Intergenerational Interaction Type of Neural Network: Self Organizing Map Differential Convolutional Neural Network (SOMDiffCNN), working under the guidance of Prof. Dr. Mutlu AVCI. Her research reflects her dedication to advancing computational methodologies and contributing to the field of computer science.

# APPENDICES

The Publication about the Thesis is attached