



İSKENDERUN TEKNİK

ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

**DOKTORA
TEZİ**

**BİYOSİNİFLANDIRMA VE ANALİZ
ARACI TASARIM VE
GERÇEKLEŞTİRİMİ**

Hüseyin ATASOY

**ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI**

OCAK 2024



**BİYOSINIFLANDIRMA VE ANALİZ ARACI TASARIM VE
GERÇEKLEŞTİRİMİ**

Hüseyin ATASOY

**DOKTORA TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**İSKENDERUN TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

OCAK 2024

Hüseyin ATASOY tarafından hazırlanan “BİYOSİNİFLANDIRMA VE ANALİZ ARACI TASARIM VE GERÇEKLEŞTİRİMİ” adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile İskenderun Teknik Üniversitesi Elektrik-Elektronik Mühendisliği Anabilim Dalında DOKTORA TEZİ olarak kabul edilmiştir.

Danışman: Doçent Dr. Yakup KUTLU

Bilgisayar Mühendisliği Anabilim Dalı, İskenderun Teknik Üniversitesi
Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

.....
.....

Başkan: Doçent Dr. Yalçın İŞLER

Biyomekanik Anabilim Dalı, İzmir Kâtip Çelebi Üniversitesi
Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

.....
.....

Üye: Doçent Dr. Sertan ALKAN

Bilgisayar Mühendisliği Anabilim Dalı, İskenderun Teknik Üniversitesi
Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

.....
.....

Üye: Doçent Dr. Oğuzhan AKGÖL

Elektrik Elektronik Mühendisliği Anabilim Dalı, İskenderun Teknik Üniversitesi
Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

.....
.....

Üye: Dr. Öğr. Üyesi Mustafa YENİAD

Bilgisayar Mühendisliği Anabilim Dalı, Ankara Yıldırım Beyazıt Üniversitesi
Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

.....
.....

Tez Savunma Tarihi: 27/01/2024

Jüri tarafından kabul edilen bu tezin Doktora Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

.....

Doçent Dr. Ersin BAHÇECİ

Lisansüstü Eğitim Enstitüsü Müdürü

ETİK BEYAN

İskenderun Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmada;

- ☒ Tez üzerinde Yükseköğretim Kurulu tarafından hiçbir değişiklik yapılamayacağı için tezin bilgisayar ekranında görüntülendiğinde asıl nüsha ile aynı olması sorumluluğunun tarafıma ait olduğunu,
 - ☒ Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
 - ☒ Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
 - ☒ Tez çalışmada yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
 - ☒ Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
 - ☒ Bu tezde sunduğum çalışmanın özgün olduğunu,
- bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Hüseyin ATASOY

27/01/2024

BİYOSINIFLANDIRMA VE ANALİZ ARACI TASARIM VE GERÇEKLEŞTİRİMİ
(Doktora Tezi)

Hüseyin ATASOY

İSKENDERUN TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

Ocak 2024

ÖZET

Şekil veya görüntülerin tanınması/sınıflandırılması problemi uzun yıllardır üzerinde çalışılan bir konudur. Biyolojik canlı türleri, tomografi, röntgen görüntüleri gibi farklı disiplinlerde kullanılan yöntemlerle elde edilen görüntülerin analizi, farklı disiplinlerde çalışan araştırmacıların kendi disiplinlerinin parçası olmayan görüntü işleme yöntemleri ile ilgili teknik bilgi sahibi olmasını gerektirebilmektedir. Bu çalışma kapsamında, farklı disiplinlerde kullanılabilecek, kullanımı kolay, konvolüsyonel yapay sinir ağı tabanlı bir öznitelik çıkarım ve işleme araç seti tasarımı ve implementasyonu yapılmıştır. Hazırlanan yazılım kütüphanesi ve kullanıcı arayüzü, farklı disiplinlerle alakalı popüler veri setleri üzerinde test edilmiştir.

Ön eğitilmiş konvolüsyonel yapay sinir ağları kullanılarak farklı veri setleri üzerinde yapılan testler, transfer öğrenimi konsepti ile ön eğitilmiş konvolüsyonel yapay sinir ağlarından öznitelik çıkarmak ve işlemek amacıyla geliştirilen araç setinin, başarı ile ayırt edici öznitelikler çıkarabildiğini göstermiştir.

Sunulan araç setinin gerçekleştiriminde, fonksiyonların mümkün olduğunca hızlı çalışabilmesi için, işlemcilerin, çok çekirdekliklik, "tek komut, çoklu veri" gibi çeşitli teknolojileri de kullanılmıştır. Yazılan kodlar optimize edilerek ve işlemci çekirdekleri üzerinde paralelleştirme yapılarak, işleme hızında büyük bir artış sağlanmıştır. Implementasyon esnasında karşılaşılan ortak kaynak kullanımı gibi problemler için çözümler sunulmuştur.

Araç seti kullanılarak, derin konvolüsyonel yapay sinir ağlarında, katman derinlikleri ve özniteliklerin betimleyicilikleri arasındaki ilişkiler de incelenmiştir. Farklı katmanlardan alınan özniteliklerin, öznitelikler ve sınıflar arasındaki korelasyonlar temel alınarak alt setlere indirgendiklerinde, ağın tamamından alınan çıkışlardan daha başarılı sonuçlar verebildiği gösterilmiştir.

Anahtar Kelimeler : Biyosınıflandırma, yapay zeka, görüntü işleme, derin öğrenme

Sayfa Adedi : 140

Danışman : Doçent Dr. Yakup KUTLU

BIO-CLASSIFICATION AND ANALYSIS TOOL DESIGN AND IMPLEMENTATION
(Ph.D. Thesis)

Hüseyin ATASOY

ISKENDERUN TECHNICAL UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

January 2024

ABSTRACT

The problem of recognizing/classifying shapes or images is a subject that has been studied for many years. The analysis of images obtained in different disciplines such as biological species, tomography, x-ray images may require researchers working in different disciplines to have technical knowledge about image processing methods that are not parts of their own discipline. Within the scope of this study, an easy-to-use, convolutional neural network-based feature extraction and processing toolset that can be used in different disciplines is designed and implemented. The library and the user interface are tested on popular data sets related to different disciplines.

Tests on different data sets using pre-trained convolutional neural networks show that the tools developed to extract and process features obtained using convolutional neural networks with the concept of transfer learning can successfully extract distinctive features.

Various technologies of processors such as multi-core, "single instruction, multiple data" are also used in order for the functions to run as fast as possible, in the implementations of the presented toolset. The processing speed is increased drastically by optimizing codes and making parallelization on the processor cores. Solutions are also presented for problems such as shared resource utilization, encountered during implementation.

Using the toolset, relations between layer depths and distinctiveness of features in deep convolutional neural networks are also investigated and it is shown that features obtained from different layers when reduced to sub-feature sets based on correlations between features and classes, can provide better results than the outputs of the entire network.

Key Words : Bio-classification, artificial intelligence, image processing, deep learning

Page Number : 140

Supervisor : Assoc. Prof. Dr. Yakup KUTLU

TEŞEKKÜR

Tez çalışmam boyunca bilgi birikimi ve deneyimleriyle beni destekleyen değerli tez danışman hocam sayın Doçent Dr. Yakup KUTLU'ya,

Çalışma hayatımın büyük bir kısmında aynı ofiste çalıştığımız değerli arkadaşım Kadir TOHMA'ya

Ve bu süreçte beni destekleyen aileme,

Teşekkür ederim.



İÇİNDEKİLER

	Sayfa
ÖZET.....	iv
ABSTRACT	v
TEŞEKKÜR	vi
İÇİNDEKİLER.....	vii
ÇİZELGELERİN LİSTESİ	x
ŞEKİLLERİN LİSTESİ.....	xiv
SİMGELER VE KISALTMALAR.....	xv
1. GİRİŞ	1
2. MATERYAL VE YÖNTEM	6
2.1. Kullanılan Veri Serileri.....	6
2.1.1. Flavia yaprak görüntüleri veri seti	6
2.1.2. IQ-OTH_NCCD tomografi görüntüleri veri seti	7
2.1.3. 102flowers çiçek görüntüleri veri seti	8
2.1.4. QUT (Queensland University of Technology) balık veri seti	9
2.1.5. AppleScabFDs elma görüntüleri veri seti.....	9
2.2. Konvolüsyonel Yapay Sinir Ağları	10
2.2.1. Giriş katmanı	11
2.2.2. Konvolüsyon katmanı.....	11
2.2.3. ReLU katmanı.....	12
2.2.4. Havuzlama	13
2.2.5. Tam bağlı katman	14
2.2.6. Softmax.....	14
2.3. Öznitelik Çıkarım Yöntemleri	14
2.3.1. Fourier serileri	14
2.3.2. Konvolüsyonel Yapay Sinir Ağı Tabanlı Öznitelik Çıkarımı	16
2.4. Sınıflandırma Yöntemleri	18
2.5. Performans Ölçümü	18
2.5.1. k-parçalı çapraz doğrulama.....	19
2.6. Yazılım Geliştirme Ortamı	20
2.6.1. Paralel kod yürütme desteği.....	21
2.6.2. Karşılıklı dışlayıcılar	22
2.6.3. Temel doğrusal cebir altprogramları (BLAS) kütüphaneleri.....	22

2.6.4. Düzenli ifadeler	23
3. İMPLEMENTASYON	24
3.1. Konvolüsyonel Yapay Sinir Ağı Kütüphanesi; CeNiN	24
3.2. Konvolüsyonel Yapay Sinir Ağları Öznitelik Çıkarım Araçları (CNNFET) ..	27
3.2.1. Öznitelik çıkarımı	27
3.2.2. Öznitelik normalizasyonu	29
3.2.3. Öznitelik seçimi	30
3.2.4. Farklı dosya formatlarına aktarım	31
3.2.5. Sınıflandırma	32
3.3. Arayüz Uygulaması İçerisindeki Sınıflar	33
3.4. Dosya Formatları	34
3.5. Bellek Kullanımı, Performans İyileştirmeleri ve Paralelleştirme.....	35
3.5.1. Bellek kullanımı.....	36
3.5.2. Konvolüsyon işleminin hızlandırılması.....	36
3.5.3. Paralelleştirme	38
4. DENEYSEL ÇALIŞMALAR	40
4.1. Fourier Serileri ve Parametrelerin Etkileri	40
4.2. CNNFET ile Ağın Tamamının Öznitelik Çıkarıcı Olarak Kullanıldığı Çalışmalar	42
4.2.1. Flavia yaprak görüntüleri veri seti	42
4.2.2. IQ-OTH_NCCD tomografi görüntüleri veri seti	43
4.2.3. 102flowers çiçek görüntüleri veri seti	44
4.2.4. QUT balık veri seti	45
4.2.5. AppleScabFDs elma görüntüleri veri seti.....	46
4.3. Konvolüsyon İmplementasyonlarının ve Paralelleştirmenin Etkileri.....	46
4.4. Farklı Derinliklerde Elde Edilen Özniteliklerin ve Sonuçlara Etkilerinin Analizi	47
4.4.1. Flavia yaprak görüntüleri veri seti	48
4.4.2. IQ-OTH_NCCD tomografi görüntüleri veri.....	51
4.4.3. 102flowers çiçek görüntüleri veri seti	54
4.4.4. QUT balık görüntüleri veri seti.....	58
4.4.5. AppleScabFDs elma görüntüleri veri seti.....	61
5. SONUÇ VE ÖNERİLER	67
EK A: CeNiN Kütüphanesi – Katmanların Sınıf Diyagramları	72
EK B: CeNiN Kütüphanesi – Diğer Sınıfların Sınıf Diyagramları	73

EK C: CeNiN Kütüphanesi – Kaynak Kodlar.....	74
EK D: CNNFET – Kaynak Kodlar	97
KAYNAKLAR.....	134



ÇİZELGELERİN LİSTESİ

Çizelge	Sayfa
Çizelge 2.1. Flavia veri setinde Fourier serileri parametre seçimi için kullanılan sınıflar	7
Çizelge 2.2. Örnek bir karmaşıklık matrisi	19
Çizelge 3.1. “.cenin” dosya formatı	25
Çizelge 4.1. N (terim sayısı) ve M (örnek sayısı) parametreleri 2 ile 25 arasında değerler aldığıda elde edilen sınıflandırma başarıları (%)	41
Çizelge 4.2. Flavia veri setinde elde edilen karmaşıklık matrisi	43
Çizelge 4.3. IQ-OTH_NCCD veri setinde elde edilen karmaşıklık matrisi	43
Çizelge 4.4. 102flowers veri setinde elde edilen karmaşıklık matrisi	44
Çizelge 4.5. QUT veri setinde elde edilen karmaşıklık matrisi	45
Çizelge 4.6. AppleScab veri setinde elde edilen karmaşıklık matrisi.....	46
Çizelge 4.7. Konvolüsyon katmanları, derinlikleri ve çıkış sayıları.....	47
Çizelge 4.8. Flavia veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	48
Çizelge 4.9. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle Flavia veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları .	50
Çizelge 4.10. Flavia veri setinde 14. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi	51
Çizelge 4.11. IQ-OTH_NCCD veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	52
Çizelge 4.12. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle IQ- OTH_NCCD veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları.....	53
Çizelge 4.13. IQ-OTH_NCCD veri setinde 11. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi	54
Çizelge 4.14. 102flowers veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	55
Çizelge 4.15. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle 102flowers veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları	56
Çizelge 4.16. 102flowers veri setinde 14. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi	57

Çizelge	Sayfa
Çizelge 4.17. QUT veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	58
Çizelge 4.18. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle QUT veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları	59
Çizelge 4.19. QUT veri setinde 7. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi	60
Çizelge 4.20. AppleScabFDs veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	61
Çizelge 4.21. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle AppleScabFDs veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen başarılar	63
Çizelge 4.22. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle AppleScabFDs veri setinde 5 ve 2 parçalı çapraz doğrulama yapılarak elde edilen başarılar	65
Çizelge 4.23. AppleScab veri seti, 9. katmandan çıkarılan öznitelikler ve 2 parçalı çapraz doğrulama ile sınıflandırıldığında elde edilen karmaşıklık matrisi.....	65

ŞEKİLLERİN LİSTESİ

Şekil	Sayfa
Şekil 2.1. Flavia veri seti içerisinde seçilen 10 çeşit yaprak türü	7
Şekil 2.2. IQ-OTH_NCCD veri setinden bazı örnekler	8
Şekil 2.3. 102flowers veri setinden bazı örnekler	8
Şekil 2.4. QUT veri setinden bazı örnekler	9
Şekil 2.5. AppleScabFDs veri setinden bazı örnekler	10
Şekil 2.6. VGG-f modeli genel yapısı (katmanlar üzerinde yazılan boyutlar, katman çıkışlarının boyutlarını temsil etmektedir)	11
Şekil 2.7. Solda ve sağda sırasıyla sigmoid ve hiperbolik tanjant	13
Şekil 2.8. ReLU fonksiyonu	13
Şekil 2.9. 2x2 boyutlu ve 2x2 kaydırmalı maksimum havuzlama örneği	14
Şekil 2.10. Flavia veri setindeki bir örnek görüntü ve bu görüntünün konturlarının θ ve ρ değerleri	16
Şekil 2.11. Önceden eğitilmiş bir konvolüsyonel yapay sinir ağının kullanımı	17
Şekil 2.12. Kodların .NET çatısı altında ortak dile derlenmesi ve çalıştırılması	21
Şekil 3.1. CNNFET araç seti	28
Şekil 3.2. Konvolüsyon filtrelerinin görselleştirilmesi	29
Şekil 3.3. Öznitelik normalizasyonu	30
Şekil 3.4. Öznitelik seçimi	31
Şekil 3.5. Farklı dosya formatlarına veri aktarımı	32
Şekil 3.6. CNNFET ile sınıflandırma	33
Şekil 3.7. Örnek bir .m çıktısı şablonu	34
Şekil 3.8. Örnek bir .arff çıktısı şablonu	35
Şekil 3.9. Chetlur ve arkadaşları tarafından önerilen konvolüsyon indirgeme metodu (Chetlur ve ark., 2014)	37
Şekil 4.1. Test edilen N ve M değerleri için elde edilen sınıflandırma başarıları ve en yüksek sınıflandırma başarısını (%98,5) veren en küçük N ve M değerleri ..	40
Şekil 4.2. Flavia veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	49
Şekil 4.3. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle Flavia veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları ..	50

Şekil**Sayfa**

Şekil 4.4. IQ-OTH_NCCD veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	52
Şekil 4.5. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle IQ-OTH_NCCD veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları	54
Şekil 4.6. 102flowers veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	55
Şekil 4.7. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle 102flowers veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları	57
Şekil 4.8. QUT veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	59
Şekil 4.9. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle QUT veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları .	60
Şekil 4.10. AppleScabFDs veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri	62
Şekil 4.11. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle AppleScabFDs veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları	64

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler

Açıklamalar

Kısaltmalar

Açıklamalar

ARFF

Attribute Relationship File Format / Öznitelik İlişki
Dosya Formatı

BLAS

Basic Linear Algebra Subprograms / Temel Doğrusal
Cebir Altprogramları

CNN

Convolutional Neural Network / Konvolüsyonel
Yapay Sinir Ağı

CNNFET

CNN Feature Extraction Tools / CNN Öznitelik
Çıkarım Araçları

EEG

Elektroensefalografi

GeMM

Generalized Matrix Multiplication / Genelleştirilmiş
Matris Çarpımı

RAM

Random Access Memory / Rastgele Erişimli Bellek

Regex

Regular expression / Düzenli ifade

ReLU

Rectified Linear Unit / Doğrultulmuş Doğrusal Birim

1. GİRİŞ

Şekil veya görüntü tanıma / sınıflandırma konusu, uzun yıllardır üzerinde çalışılan bir konudur. Benzer morfolojilere sahip yaprak veya hayvan türlerinin şekilleri üzerinden sınıflandırılması, görüntülerde yer alan nesne veya şekillerin tanınması, tıbbi görüntülerden çıkarımlar yapılması gibi değişik disiplinlerde, konu üzerinde yapılan çok sayıda çalışma mevcuttur. Şekiller bilgisayar ortamına aktarılırken genelde fotoğraflar kullanılmakta olduğundan, fotoğrafların görüntü işleme teknikleri ile işlenerek şekilleri meydana getiren nokta bulutlarına çevrilmesi için kullanılan çok sayıda görüntü işleme metodu da konunun kapsamı dahilindedir. Fotoğraflar farklı ortamlarda, farklı açılarla ve farklı uzaklıklardan çekilebildiğinden, yöntemlerin yeniden boyutlandırma, döndürme ve öteleme işlemlerinden etkilenmemesi gerekmektedir.

Literatürde, yeniden boyutlandırma, döndürme ve öteleme işlemlerinin olumsuz etkilerini azaltma, renkli görüntülerden şekil çıkarım kalitesini arttırma, tanıma / sınıflandırma başarımını arttırma gibi farklı amaçlara odaklanılarak önerilen çalışmalar mevcuttur. Bu çalışmalar genelde tanımlayıcı noktalar bulmaya yönelik olarak çalışmaktadır (Bay, Tuytelaars, Van Gool, 2006; Rublee, Rabaud, Konolige, Bradski, 2011).

Sınıflandırma başarısını en çok etkileyen etmenlerden bir tanesi şekillerin temsidir. Bir şeklin temsili, iskelet tabanlı, bölge tabanlı ve sınır tabanlı olabilmektedir.

İskelet tabanlı temsil metotlarında genelde orta eksenler kullanıldığından, bu tip temsillerde temsil edilmesi amaçlanan nesnelerin bir kısmı kapalı / görünmüyor olsa dahi bu nesnelerin başarılı bir şekilde temsili mümkün olabilmektedir (Ling ve Jacobs, 2007).

Bölge tabanlı temsil yöntemlerinde ise görüntülerden bölgesel öznitelikler çıkarılması amaçlanır. Zernike momentleri veya Hu momentleri kullanılarak geliştirilen temsil yöntemleri bu yöntemlere örnek olarak verilebilir (Kim, Kim, Sim, Oh, 2000; Mei ve Androutsos, 2009; Das, Chakraborty, Mitra, Maiti, Ray, 2013).

Sınır tabanlı metotlarda bir şeklin temsili için yalnızca şeklin sınırlarını belirleyen eğriler kullanılır. Fourier tanımlayıcıları, çok ölçekli fraktal boyut gibi yöntemlerde bir şekil, şeklin sınırları belirlenerek tanımlanır. Bu yöntemlerde sınırlar genellikle birbirlerine bağlı noktalar

kümesi olarak ele alınır (De Baets, Develder, Dhaene, Deschrijver, 2017; Wong ve Abas, 2017; López-Sampson ve Page, 2018).

Literatürde, yaprakların şekil analizi, balıkların tür sınıflandırması, yaş tayini, mikroskopik görüntülerin analizi gibi farklı problemlere çözüm sunan, farklı disiplinlerde işlev gören yöntem ve araçlar da mevcuttur (Gyori, Venkatachalam, Thiagarajan, Hsu, Clement, 2014; Savitzky, Zeltmann, Hughes, Brown, 2018; Schneider, Weiss, Ruhe, Jung, Roggenbuck, Stohwasser, 2019, Busato, Servi, Volpe, Furferi, Scorianz, 2023). Ancak bu araçlar genelde probleme özgü olarak geliştirilmiştir (Końca, Lankoff, Banasik, Lisowska, Kuszewski, Gózdź, Wojcik, 2003; Olive ve Banáth, 2006; Spampinato, Giordano, Di Salvo, Chen-Burger, Fisher, Nadarajan, 2010; Chen, Sun, Shang, 2017; Jung, 2021).

Sadafi ve arkadaşları kan hücrelerinin morfolojik analizlerini kolaylaştırmak amacıyla RedTell olarak isimlendirdikleri bir araç geliştirmişlerdir (Sadafi, Bordukova, Makhro, Navab, Bogdanova, Marr, 2023). Araç ile bölütleme, veri işaretleme gibi işlemler yapılabilmektedir. Bölütleme işlemi için araç içerisinde önceden eğitilmiş bir bölgesel konvolüsyonel yapay sinir ağı kullanılmıştır.

Razeghi ve arkadaşları, kardiyovasküler araştırmalar, tanı ve tedavilerine yönelik makine öğrenimi ve simülasyon yaklaşımlarını uygulamak için görüntü işleme ve bilgisayarla görme yöntemlerini içeren CemrgApp isimli bir araç seti geliştirmiştir (Razeghi, Solís-Lemus, Lee, Karim, Corrado, Roney, Niederer, 2020).

Corradini ve arkadaşları, Spektral açı haritalama tekniği ile Fourier dönüşümlü kızılötesi spektroskopi görüntülerini sınıflandıran bir R paketi yazıp araştırmacıların kullanımına sunmuştur (Corradini, Beriot, Huerta-Lwanga, Geissen, 2021).

Bir başka çalışmada Temiz, görüntülerde süper çözünürlüklü yeniden yapılandırma işlemleri için derin öğrenme tabanlı DeepSR isimli bir araç yayınlamıştır (Temiz, 2023).

González-Ruiz ve Fernández, çalışmalarında, 3 boyutlu elektron mikroskopları ile elde edilen görüntüler için yapılar korunarak gürültü gidermek amacıyla bir yazılım aracı hazırlamış ve yayınlamıştır (González-Ruiz ve Fernández, 2023).

Sanders ve arkadaşları, medikal görüntülerde analizler yapılabilmesini kolaylaştırmak için derin yapay sinir ağları araçları içeren DLAE isimli bir uygulama yayınlamışlardır (Sanders, Fletcher, Frank, Liu, Johnson, Zhou, Ma, 2019). Uygulamalarını MRI (manyetik rezonans görüntüleme) görüntülerinde test etmiş ve başarılı sonuçlar raporlamışlardır.

Benzer şekilde, araştırmacıların hizmetine sunulan, farklı problemlere odaklanmış çok sayıda araç seti ve yazılım mevcuttur. Literatürde, önceden eğitilmiş konvolüsyonel yapay sinir ağlarının öznetelik çıkarıcı veya doğrudan sınıflandırıcı olarak kullanıldığı çalışmalar da bulunmaktadır.

Varshni ve arkadaşları, 2019 yılında yayınladıkları çalışmada, zatürre tanısını röntgen görüntülerinden yapmayı amaçlamıştır (Varshni, Thakra, Agarwal, Nijhawan, Mittal, 2019). Görüntüler 224x224 boyutlarına indirgendikten sonra önceden farklı görüntülerle eğitilmiş bir derin yapay sinir ağından geçirilmiştir. Son katmandan hemen önce elde edilen vektör öznetelik vektörü olarak değerlendirilmiş ve destek vektör makineleri yöntemiyle sınıflandırma işlemi gerçekleştirilmiştir. Araştırmacılar farklı derin ağ modelleri kullanarak sonuçlar almış ve sonuçları karşılaştırmıştır.

Hosseini ve arkadaşları, lösemi tanısını mikroskopik görüntülerden otomatik olarak yapabilmek için, mobil telefonlarda çalışabilecek hafif bir konvolüsyonel yapay sinir ağı kullanmışlardır (Hosseini, Eshraghi, Taami, Sadeghsalehi, Hoseinzadeh, Ghaderzadeh, Rafiee, 2023). Çalışmalarında, kullandıkları görüntüleri bazı ön işlemlerden geçirip gereksiz bileşenleri kaldırdıktan sonra, üç farklı hafif konvolüsyonel ağ üzerinde sonuçları karşılaştırmış ve en etkin ağı seçip mobil uygulama implementasyonunu yapmışlardır.

Wang ve arkadaşları, elektroensefalografi (EEG) sinyalleri ile depresyon arasında bir ilişki yakalamaya çalışmıştır (Wang, Kang, Huo, Chen, Song, Zhang, 2023). Çalışmalarında konvolüsyonel yapay sinir ağlarını yine bir öznetelik çıkartım aracı olarak kullanmışlardır. Ancak EEG sinyallerini önce 2 boyutlu bir görüntü haline getirip, bir derin yapay sinir ağı modelini eğitmişlerdir. Çalışmanın sonunda, konvolüsyonel yapay sinir ağlarının genelleme başarısı sayesinde, depresyon ile bazı beyin bölgeleri arasında güçlü bir ilişki yakalandığı raporlanmıştır.

Khan ve arkadaşları, diyabetik retinopatinin görüntülerden tespiti için ön eğitilmiş konvolüsyonel yapay sinir ağı modelleri kullanmışlardır (Khan, Abbas, Rizvi, 2019). Ön eğitilmiş ağlarda %94,49'a kadar başarılı sınıflandırılabilen bu rahatsızlık, kendi geliştirdikleri ağ yapısında %98,15'lik bir başarı ile sınıflandırılmıştır.

González- Lozoya ve arkadaşları bir ön eğitilmiş konvolüsyonel yapay sinir ağı kullanarak, yüz ifadelerini ve mikro-ifadeleri tanımayı amaçlayan bir çalışma yapmıştır (González-Lozoya, Calleja, Pellegrin, Escalante, Medina, Benitez-Ruiz, 2020). Yaptıkları çalışmada konvolüsyonel ağlarla, 6 evrensel yüz ifadesi içeren çok sayıda görüntünün %92'sini doğru sınıflandırmışlardır.

Alzheimer tanısı ve aşamasının belirlenmesi için önceden eğitilmiş yapay sinir ağı modellerinin kullanıldığı bir diğer çalışmada Nawaz ve arkadaşları, çıkardıkları öznitelikleri destek vektör makineleri, k-en yakın komşu ve rastgele orman algoritmaları ile en yüksek %99,21'lik bir sınıflandırma başarıları raporlamışlardır (Nawaz, Maqsood, Afzal, Aadil, Mehmood, Rho, 2021).

Vijayalakshmi, mikroskobik görüntülerden sıtma teşhisi için bir önceden eğitilmiş konvolüsyonel ağ ve destek vektör makineleri yöntemini kullanmıştır. Bu yaklaşımla kullanılan görüntü setinde elde edilen sınıflandırma başarıları %93,1 olmuştur.

Konvolüsyonel yapay sinir ağları, ayırt edici öznitelikler yakalama becerileri nedeniyle, günümüzde farklı disiplinlerde oldukça popüler hale gelmiştir. Ancak çoğu durumda programlama becerisi ile birlikte, yeni bir ağ modeli tasarımına gereksinim duyulacaksa, teknik bilgi ve beceri de gerektirmektedir. Bu nedenle, görüntü işleme, makine öğrenmesi gibi konulara nispeten uzak disiplinlerde çalışan araştırmacılar için, kullanımı kolay, bir arayüz üzerinden yönetilebilen, kurulum ve ek yazılım bağımlılığı gerektirmeden çalışabilen bir yazılım aracının araştırma faaliyetlerine yardımcı olabileceği öngörülmüştür. Bu kapsamda temel katmanların implementasyonları içeren açık kaynak kodlu bir yapay sinir ağı kütüphanesi ve bu kütüphaneyi kullanan, kurulum gerektirmeyen bir Windows uygulama arayüzü yazılmıştır.

Çalışma kapsamında Fourier serileri metodunun da implementasyonu BioMorph (Kutlu ve Turan, 2018) isimli bir yazılıma modül olarak yazılmış ve kullanıma sunulmuştur. Yöntemin

ve implementasyonun etkinliĐinin incelenmesi amacıyla bu yntem ile yaprak grntleri zerinde alıřmalar yapılmıř ve elde edilen sonular deĐerlendirilmiřtir.

alıřma kapsamında n eĐitilmiř bir konvolsyonel yapay sinir aĐının eĐitim ařamasında grmediĐi grntler zerinde ayırt edici znitelikler bulma bařarısı da deĐerlendirilmiřtir.

alıřmanın devamında, implementasyon ařamasında karřılařılan performans problemleri, yapılan iyileřtirmeler, paralelleřtirme esnasına yařanan bellek problemleri ve sunulan zmler de irdelenmiřtir.

Sonraki blmde, testlerde kullanılan veri setleri, yntemler, yazılım geliřtirme ortamı hakkında bilgiler verilmiřtir. Daha sonra implementasyon ařamasında tasarlanan sınıflar, yazılan ktphane ve ara seti tanıtılmıř, karřılařılan sorunlar ve nerilen zmler detaylandırılmıřtır. Bir sonraki blmde, veri setleri zerinde yapılan deneysel alıřmalar ve elde edilen sonular paylařılmıřtır. Daha sonra veri setleri zerinde aĐların derinlikleri ile zniteliklerin temsil yetenekleri arasındaki iliřkiler incelenmiř ve farklı katmanlardan alınan ıkıřların sınıflandırma bařarılarına etkileri deĐerlendirilmiřtir.

Son blmde, geliřtirilen araların etkinliĐi, elde edilen sonular ıřıĐında tartıřılmıř ve neriler yazılmıřtır.

2. MATERİYAL VE YÖNTEM

Yazılan araç seti için, kullanılacak yöntemlerin ve başarılarının ölçümü farklı veri setleri ve metotlar kullanılarak yapılmıştır. Araç setine dahil edilmek üzere öncelikle kontur tabanlı bir yöntem olan Fourier serileri ve daha sonra konvolüsyonel yapay sinir ağları incelenmiştir. Yazılan modüller ve yapılan testlerin ardından sınıflandırma aracına yalnızca konvolüsyonel yapay sinir ağlarının dahil edilmesi kararlaştırılarak, araç seti bu modern metot üzerinde şekillendirilmiş ve Fourier serileri için yapılan implementasyon araç seti ile birlikte daha sonra BioMorph (Kutlu ve Turan, 2018) isimli yazılıma modül olarak eklenmiştir.

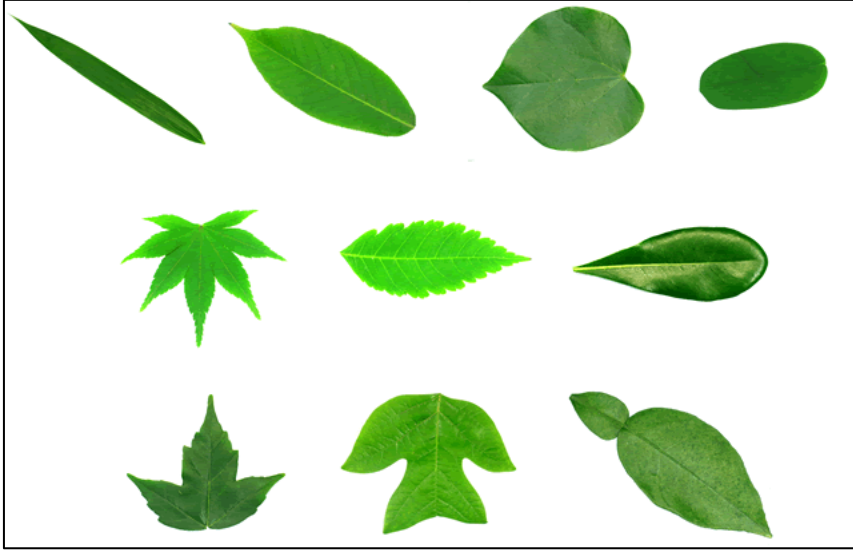
2.1. Kullanılan Veri Serileri

Geliştirilen araç setinin, farklı türlerde görüntüler içeren, farklı disiplinlerde kullanılan çok sayıda veri seti üzerinde başarılı olabileceğini göstermek için çeşitli veri setleri kullanılmıştır.

Farklı disiplinlerde farklı amaçlarla toplanmış, yaprak türleri, akciğer tomografileri, çiçek türleri, balık görüntüleri ve hastalıklı / hastaliksız elma görüntüleri içeren 5 veri seti, çalışma kapsamında kullanılmıştır.

2.1.1. Flavia yaprak görüntüleri veri seti

Fourier serileri üzerinde parametre seçimi yapılabilmesi amacıyla Flavia yaprak veri setinden (Wu, Bao, Xu, Wang, Chang, Xiang, 2007) elde edilen bir alt veri seti kullanılmıştır. Flavia, 32 tür bitki yaprağının RGB formatında görüntülerini içeren bir veri setidir. Veri seti içerisindeki bazı türler birbirlerine fazlaca benzediğinden ve çalışmanın amacı veri setinde iyi sonuçlar almaktan ziyade, parametrelerin etkilerini incelemek olduğundan, birbirine benzeyen türler atılarak, bir alt veri seti oluşturulma ihtiyacı duyulmuştur. Alt veri seti, her türde 20 olmak üzere, toplam 200 adet yaprak görüntüsünden oluşmaktadır. Seçilen yapraklar ve sınıflar sırasıyla Şekil 2.1’de ve Çizelge 2.1’de verilmiştir.



Şekil 2.1. Flavia veri seti içerisinde seçilen 10 çeşit yaprak türü

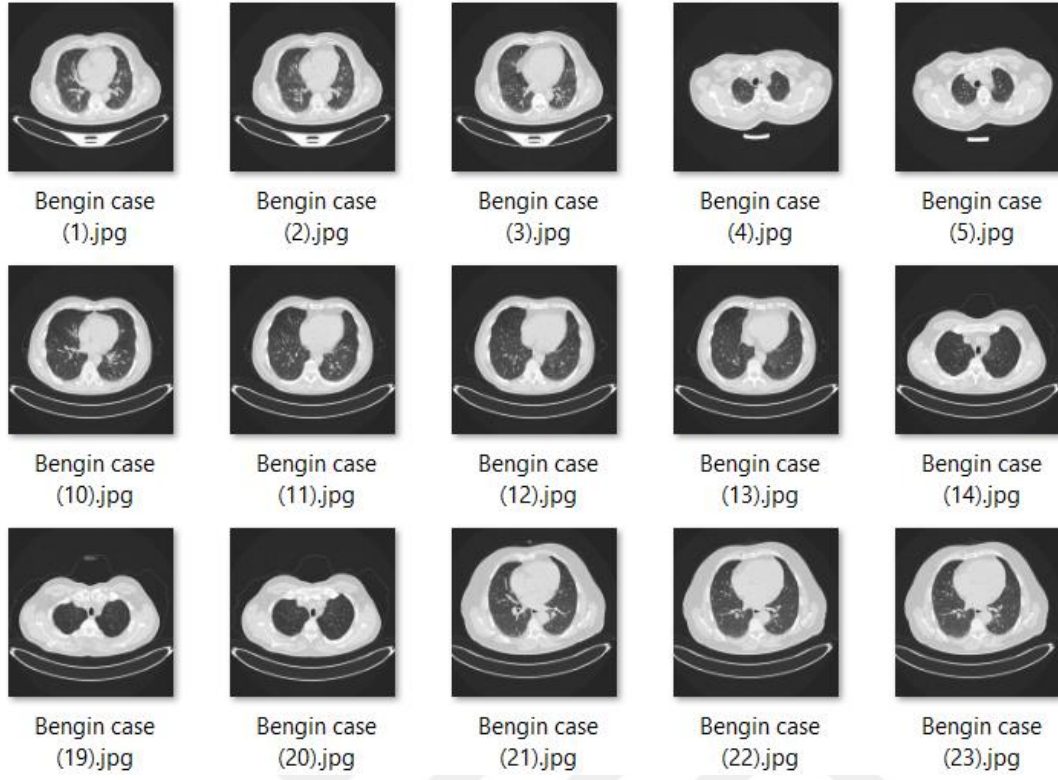
Çizelge 2.1. Flavia veri setinde Fourier serileri parametre seçimi için kullanılan sınıflar

Yaprak	Görüntü Sayısı
pubescent bamboo	10
chinese horse chestnut	10
chinese redbud	10
true indigo	10
japanese maple	10
goldenrain tree	10
japanese cheesewood	10
trident maple	10
chinese tulip tree	10
tangerine	10

Veri setinin, geliştirilen araç setinde test edilebilmesi içinse veri seti içerisindeki tüm görüntüler kullanılmıştır.

2.1.2. IQ-OTH_NCCD tomografi görüntüleri veri seti

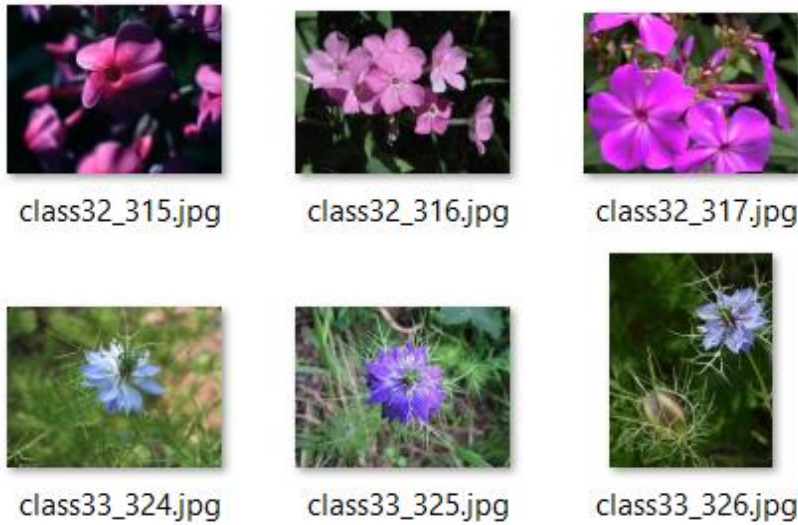
IQ-OTH_NCCD veri seti (Al-Yasriy, Al-Husieny, Mohsen, Khalil, Hassan, 2020) bir akciğer kanseri tomografileri veri setidir. İçerisinde 120'si iyi huylu, 561'i kötü huylu, 416'sı tümörsüz olmak üzere toplamda 1097 adet vakanın tomografi görüntülerinden oluşmaktadır (Şekil 2.2).



Şekil 2.2. IQ-OTH_NCCD veri setinden bazı örnekler

2.1.3. 102flowers çiçek görüntüleri veri seti

102flowers veri seti (Nilsback ve Zisseran, 2008) 102 farklı türe ait 8189 çiçek görüntüsünden meydana gelen bir veri setidir. (Şekil 2.3).

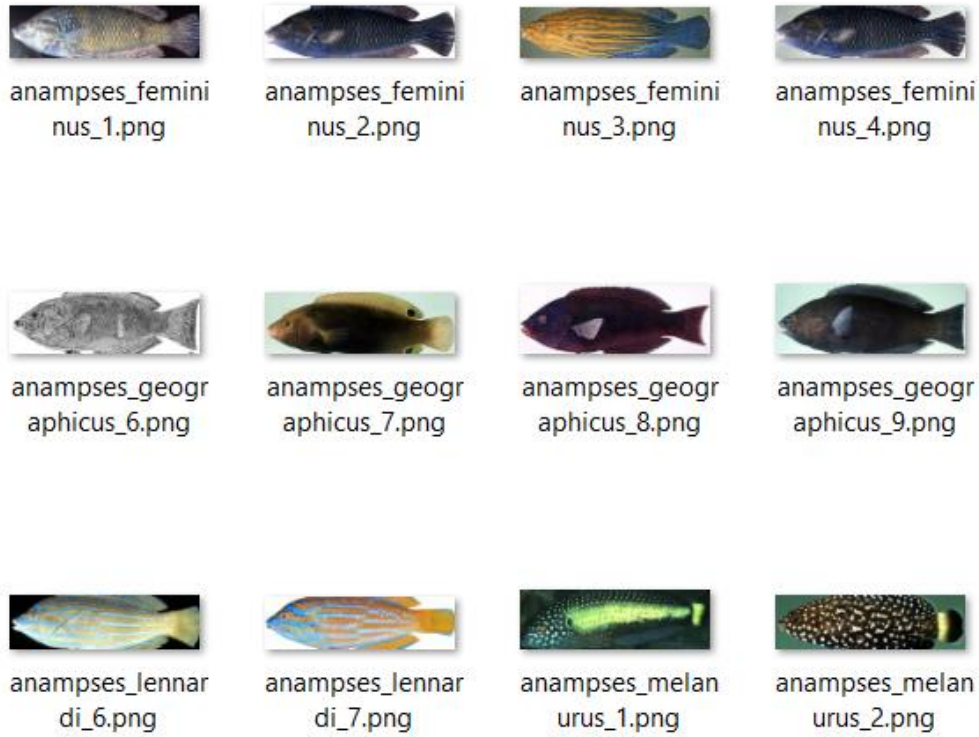


Şekil 2.3. 102flowers veri setinden bazı örnekler

Veri setinde bazı kategorilere ait çok az sayıda görüntü bulunduğundan ve sınıflandırma 10 parçalı çapraz doğrulama ile yapılacağından, her parçaya en az 10 örnek düştüğünün garantilenebilmesi için 100 örnekten daha az örnek barındıran sınıflar elenmiş ve 23 kategoride 3375 görüntü içeren bir alt veri seti elde edilmiş ve testler bu alt veri seti üzerinde yürütülmüştür.

2.1.4. QUT (Queensland University of Technology) balık veri seti

QUT (Srinivasan, 2020) seti 468 türe ait 4415 görüntü barındırmaktadır. Ancak bazı kategorilerde az sayıda görüntü yer aldığından en az 20 örnek içermeyen türler elenmiştir. Geriye 18 türden oluşan 401 görüntü içeren bir alt set kalmıştır (Şekil 2.4).



Şekil 2.4. QUT veri setinden bazı örnekler

2.1.5. AppleScabFDs elma görüntüleri veri seti

AppleScabFDs (Kodors, Lacis, Sokolova, Zhukovs, Apeinans, Bartulsons, 2021) seti, 90 adet sağlıklı ve 207 adet hastalıklı elma görüntülerine ait toplam 297 adet görüntüden oluşmaktadır. Görüntüler, doğal ortamda çekilmiş ağaçlarda duran elma ve yaprakların da

dahil olduğu görüntülerdir. Hastalıklı elma meyve ve yapraklarında siyah lekeler mevcuttur (Şekil 2.5).



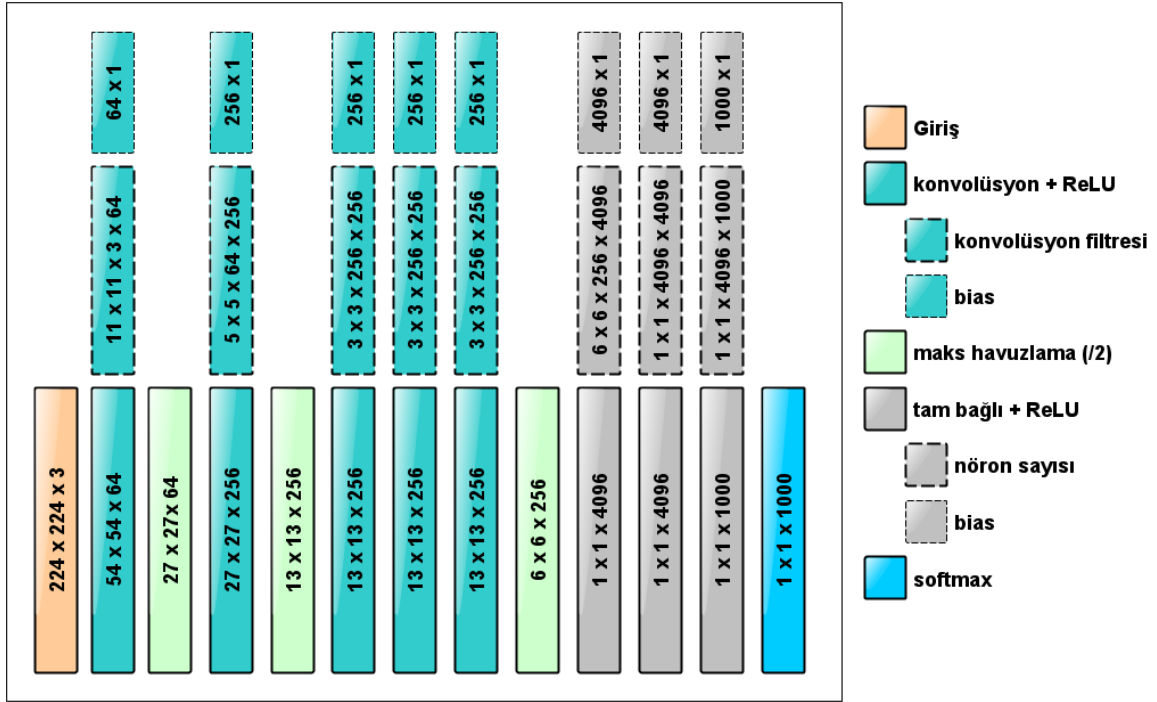
Şekil 2.5. AppleScabFDs veri setinden bazı örnekler

2.2. Konvolüsyonel Yapay Sinir Ağları

Konvolüsyonel ağların temelinde yatan fikir, Hubel ve Wiesel tarafından yayınlanan, bir kedi beyninin görsel korteksinin fonksiyonel analizine dayanmaktadır (Hubel ve Wiesel, 1962). Görme duyusu ile ilgili bu bulgulara dayanarak tasarlanan konvolüsyonel yapay sinir ağı modelleri, teknolojinin ve donanım araçlarının son 15-20 yıl içerisindeki gelişimi ile birlikte daha uygulanabilir hale gelmiş, görüntü işleme konusundaki başarılarını ispatlamış ve oldukça popüler hale gelmiştir.

Farklı çok sayıda model mevcut olmak birlikte, tez çalışmasında, başarısını kanıtlamış modellerden biri olan VGG-f modeli kullanılmıştır (Chatfield ve ark., 2014). Modelin genel yapısı Şekil 2.6’da gösterilmiştir.

Tez çalışması boyunca testler, VGG-f modelinde olan “imagenet-matconvnet-vgg-f.cenin” ön eğitilmiş ağı kullanılarak gerçekleştirilmiştir. Bu ağ, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 verisi ile eğitilmiş bir ağ olup, tez çalışması kapsamında tasarlanan araç setinin kullandığı dosya formatı olan cenin formatına çevrilmiştir. Ağın eğitimi, 1000 farklı sınıfa ait 1,2 milyon görüntü kullanılarak gerçekleştirilmiştir (Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015).



Şekil 2.6. VGG-f modeli genel yapısı (katmanlar üzerinde yazılan boyutlar, katman çıkışlarının boyutlarını temsil etmektedir)

2.2.1. Giriş katmanı

Giriş katmanı, görüntünün ağa girişinin yapıldığı katmandır. Ağa girilmesi gereken görüntünün boyutlarını ve renk derinliğini bu katman belirler. Ağa giriş olarak verilen görüntü, sonraki katmanlarda işlenebilecek boyutlara getirilmek için gerekiyorsa bu katmada yeniden boyutlandırılır.

2.2.2. Konvolüsyon katmanı

Konvolüsyon katmanı, görüntüler üzerinde gezdirilecek çok sayıda konvolüsyon filtresinin yer aldığı katmandır. Görüntü bu katmanlardan her birinden geçerken, katmanda yer alan ve değerleri eğitim aşamasında belirlenip optimize edilen konvolüsyon filtrelerinin her biri ile konvolüsyon işlemine tabi tutulur (Eşitlik 2.1).

$$F * I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j) I(x - i, y - j) \quad (2.1)$$

Ancak pratikte, ağdan her bir görüntü işlenirken, filtrelerin ters çevrilip kullanılması zaman açısından gereksiz ve tekrarlı bir maliyet oluşturacaktır. Bu nedenle pratikte, filtreler bellekte zaten ters tutulup konvolüsyon işlemi yerine çapraz korelasyon işlemi

gerçekleştirilir (Eşitlik 2.2). Ancak çalışmanın devamında bu işlem, terminolojiye uygun olarak konvolüsyon olarak anılmaya devam edecektir.

$$F \circ I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j) I(x + i, y + j) \quad (2.2)$$

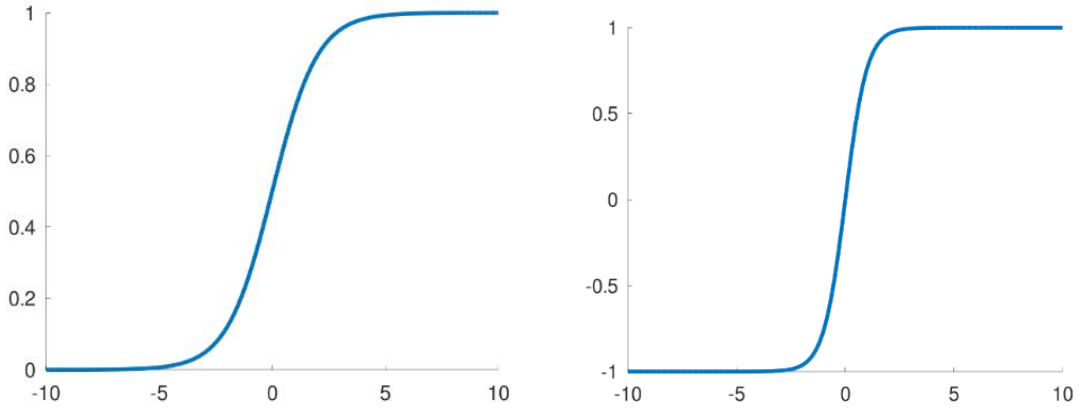
2.2.3. ReLU katmanı

ReLU (rectified linear unit), aktivasyon fonksiyonu katmanıdır. Klasik yapay sinir ağlarında sıkça kullanılan sigmoid (Eşitlik 2.3, Şekil 2.7) veya hiperbolik tanjant (Eşitlik 2.4, Şekil 2.7) gibi fonksiyonlar, derin ağlarda kullanıma uygun değildir. Konvolüsyonel derin ağlar, gradyan iniş metodu ile optimize edilirken, hataların geri yayılımı esnasında sigmoid ve hiperbolik tanjant fonksiyonları, ilk katmana doğru gradyanların çok fazla küçülmesine sebep olur. Sigmoid fonksiyonunun sıfıra yakın noktalardaki türevleri de 0'a oldukça yakındır. Benzer şekilde, 1'e yakın noktalardaki türevler de 0'a yakındır. Giriş katmanına doğru hatalar geri yayılırken, türevler giderek daha da küçülür ve sıfıra yaklaşır. Bu nedenle derin ağlarda girişe yakın katmanlarda öğrenme oldukça yavaş gerçekleşir. Veriler arasında doğrusal olmayan karmaşık ilişkilerin de yakalanabilmesi için aktivasyon fonksiyonlarının doğrusal olması da istenen bir durum değildir. ReLU (Eşitlik 2.5, Şekil 2.8), bir yandan, türevi, 0'dan büyük giriş değerleri için 1 olduğundan, gradyanın küçülüp sıfırlanmasını önlerken, diğer yandan, 0'dan küçük değerler için 0 döndürdüğünden, doğrusal olmanın dezavantajını da taşımaz.

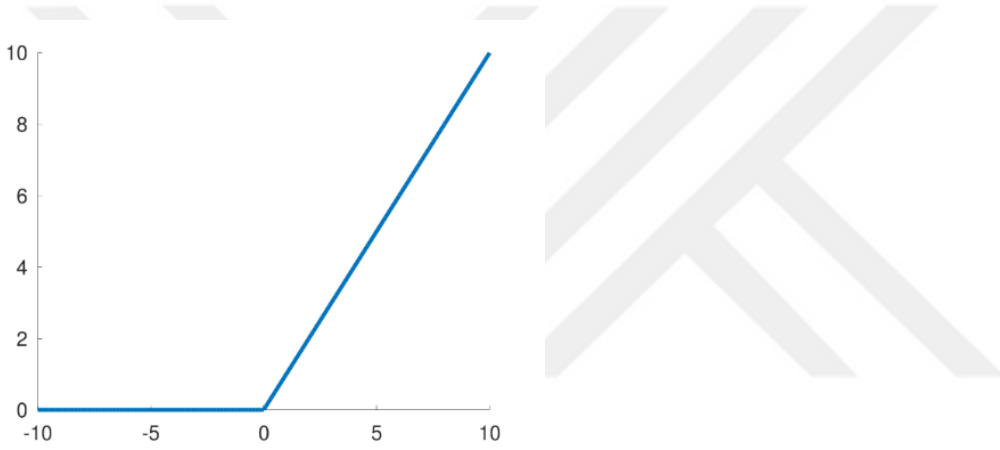
$$f(x) = \frac{1}{1+e^{-x}} \quad (2.3)$$

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.4)$$

$$f(x) = \max(0, x) \quad (2.5)$$



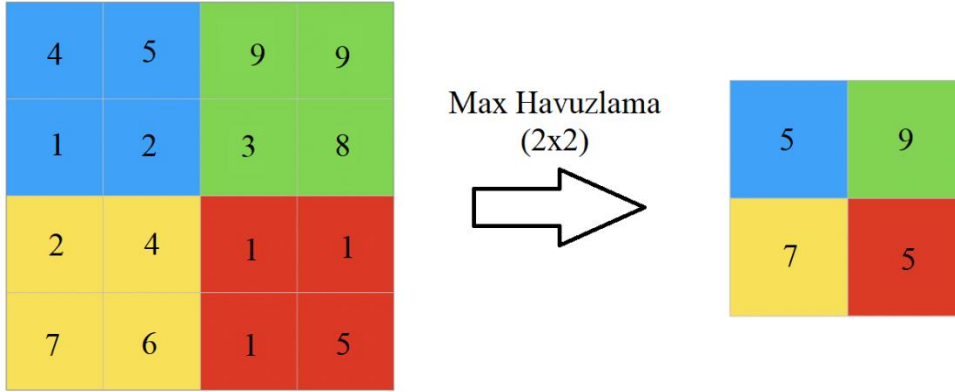
Şekil 2.7. Solda ve sağda sırasıyla sigmoid ve hiperbolik tanjant



Şekil 2.8. ReLU fonksiyonu

2.2.4. Havuzlama

Havuzlama, görüntülerde ötelemenin olumsuz etkisini azaltmak amacıyla kullanılmaktadır. Havuzlama yapılırken, görüntü $n \times n$ 'lik havuzlara bölünür ve her piksel havuzundan, maksimum, minimum, ortalama gibi değerlerinden biri seçilir. Örneğin 4×4 'lük bir maksimum havuzlama tekniği kullanıldığında, görüntüdeki her 4×4 'lük bölgeden en yüksek değere sahip piksel seçilir ve çıkışa aktarılır. Böylece görüntünün boyutu yarı yarıya düşer (Şekil 2.9).



Şekil 2.9. 2x2 boyutlu ve 2x2 kaydırmalı maksimum havuzlama örneği

2.2.5. Tam bağlı katman

Tam bağlı katman, klasik yapay sinir ağlarında olduğu gibi, katmandaki tüm nöronların sonraki katmandaki tüm nöronlarla birer bağlantısının olduğu katmanlardır. Bu katmanlar, konvolüsyon katmanları tarafından yakalanan öznitelikleri sınıflandıran klasik yapay sinir ağlarıdır.

2.2.6. Softmax

Softmax, aldığı girişleri (önceki katmanın çıkışlarını), tüm çıkışların toplamının 1 olmasını sağlayacak şekilde olasılıklara çevirerek, ağın çıkışlarını belirler (Eşitlik 2.6).

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad (2.6)$$

2.3. Öznitelik Çıkarım Yöntemleri

Tez çalışması kapsamında, öznitelik seçimi için başta Fourier serileri ve parametre seçimleri üzerinde çalışılmış, daha sonra konvolüsyonel yapay sinir ağları kullanılmıştır.

2.3.1. Fourier serileri

$[0, 2L]$ aralığında tanımlı bir periyodik fonksiyon olan $f(x)$ a_n ve b_n ile çarpılmış sonsuz sayıda sinüs ve kosinüs terimi ile ifade edilebilir (Eşitlik 2.7 – 2.10) (Weisstein, 2004).

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(n\pi x) + \sum_{n=1}^{\infty} b_n \sin(n\pi x) \quad (2.7)$$

$$a_0 = \frac{1}{L} \int_0^{2L} f(x) dx \quad (2.8)$$

$$a_n = \frac{1}{L} \int_0^{2L} f(x) \cos(n\pi x) dx \quad (2.9)$$

$$b_n = \frac{1}{L} \int_0^{2L} f(x) \sin(n\pi x) dx \quad (2.10)$$

Konturlar, bilgisayar üzerinde sürekli bir $f(x)$ fonksiyonu olarak değil, ayrık örnekler halinde tutulur. M , örneklerin sayısını temsil ederken, bu örneklerin $[0, 2L]$ aralığında tanımlı olduğu varsayıldığında L değeri $M/2$ 'ye eşit olur. Bu durumda Fourier serileri kullanılarak yeniden yapılandırılmış $f(x)$ fonksiyonu Eşitlik 2.11'de olduğu gibi ifade edilebilir.

$$f'(x) = \frac{1}{2}a_0 + \sum_{n=1}^N a_n \cos\left(\frac{n\pi x}{L}\right) + \sum_{n=1}^N b_n \sin\left(\frac{n\pi x}{L}\right), \quad x \in [0, 2L] \quad (2.11)$$

Burada N , sinüs ve kosinüs terimlerinin sayısını temsil etmektedir ve $f(x)$ fonksiyonunun $f(x)$ 'e yeterince benzemesini sağlayacak bir değerde seçilir.

Bu işlemler esnasında seçimi sonuçları doğrudan etkileyen iki önemli parametre bulunmaktadır. Bunlardan ilki, sinüs-kosinüs terimlerinin sayısı olan N değeridir. N arttıkça, $f(x)$ fonksiyonunun $f(x)$ 'e benzerliği artar. Diğer bir önemli parametre ise yeniden yapılandırma esnasında terimlerden toplanacak örnek sayısı olan M değeridir. Benzer şekilde M arttıkça $f(x)$ fonksiyonunun $f(x)$ 'e benzerliği yine artar. Ancak her iki parametrenin de mümkün olan en düşük değerde seçilmesi hem hesaplama maliyetinin düşürülmesi hem de yeniden yapılandırma işleminin sonunda üzerinde çalışılacak değerlerin sayısının azaltılması için önemlidir.

Fourier serileri kullanılırken, görüntülerdeki nesnelerin konturları sinyaller olarak ele alınır. Konturlar, görüntülerdeki nesnelerin kenarlarını temsil eden noktalar kümesidir. Bu noktalar 2 boyutludur ve her boyuttaki değerler bir zaman serisi olarak düşünülebilir.

Görüntülerdeki nesnelerin yönelim açıları veya konumları çok az değiştiğinde bile kartezyen sistemde koordinatlar tümüyle değişecektir. Ayrıca, veri setindeki yaprakların yönelim

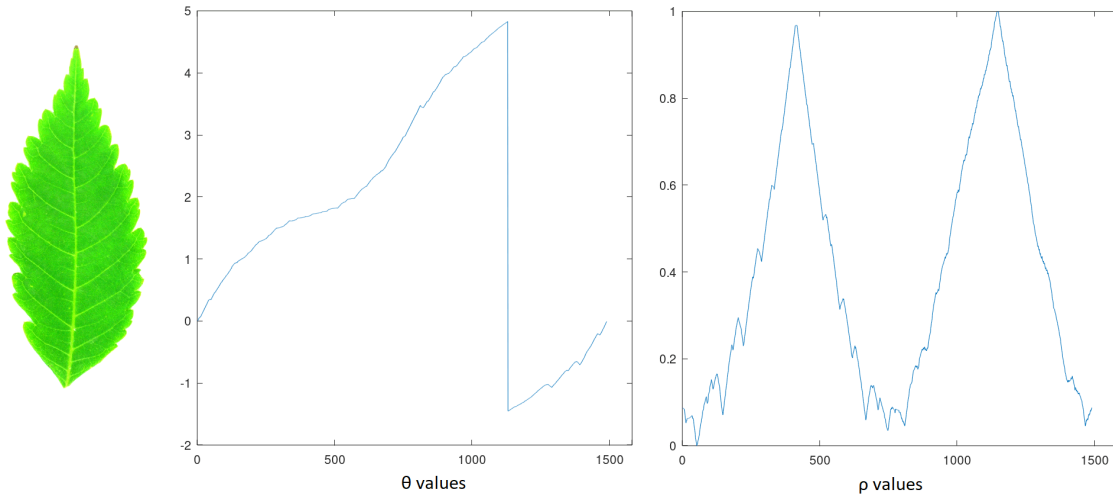
açıları, boyutları ve konumları tamamen aynı değildir. Bu nedenle kartezyen koordinatlar (x, y) polar koordinatlara (θ, ρ) dönüştürülmüştür (Eşitlik 2.12).

$$(\theta, \rho) = \left(\tan^{-1} \frac{y-y_c}{x-x_c}, \sqrt{(x-x_c)^2 + (y-y_c)^2} \right) \quad (2.12)$$

Burada x_c ve y_c konturun merkezinin kartezyen koordinatlarıdır (Eşitlik 2.13).

$$(x_c, y_c) = \left(\frac{1}{N} \sum_{n=1}^N x_n, \frac{1}{N} \sum_{n=1}^N y_n \right) \quad (2.13)$$

N değeri, nokta sayısını temsil etmektedir.



Şekil 2.10. Flavia veri setindeki bir örnek görüntü ve bu görüntünün konturlarının θ ve ρ değerleri

Veri setinden alınan bir örneğin konturunun θ ve ρ değerleri Şekil 2.10'da verilmiştir. Şekilde de görüldüğü üzere, θ değerlerinde keskin bir geçiş meydana gelmektedir. Bu keskin geçiş, eğrinin sinüs ve kosinüs terimleri ile gösterimini zorlaştırmaktadır. Ayrıca θ ve ρ değerleri tamamen bağımsız olmadığından ρ , θ kullanılmadığında da çok fazla veri kaybı olmadan konturları temsil edebilir. Bu nedenle Fourier katsayıları, yürütülen testler boyunca ρ değerleri üzerinde hesaplanmıştır.

2.3.2. Konvolüsyonel Yapay Sinir Ağı Tabanlı Öznitelik Çıkarımı

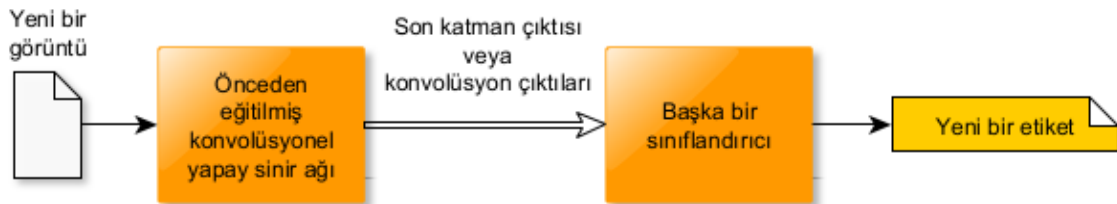
Ön eğitilmiş bir yapay sinir ağının öznitelik çıkarıcı olarak kullanılabilmesinin arkasında yatan temel dayanak, ağların öğrendiği konvolüsyon filtrelerinin ayırt edici örüntüler çıkarabilme becerisidir. Farklı ölçeklerde çok sayıda konvolüsyon filtresi içeren derin

konvolüsyonel ağlardaki filtreler, eğitim esnasında optimize edilerek ağın istenen çıktıları vermesini sağlarlar.

Eğitilmiş bir ağdan, daha önce ağa verilmemiş bir görüntü geçirildiğinde ve son katmana gelindiğinde, ağ, doğru veya yanlış, mutlaka bir sonuç verecektir. Ağdan alınacak sonuç, görüntüdeki örüntülerin aslında eğitim aşamasında hangi görüntü sınıfındaki örüntülere benzediğinin bir ölçüsü olacaktır. Dolayısıyla ağın, yeni görüntünün sınıfı için daha önce eğitilmiş olmasına ve daha önce görmediği bir görüntüde gerçekte ne yer aldığını bilmesine gerek yoktur. Ağın vereceği çıktı, yeni görüntünün daha önce eğitim aşamasında kullanılan sınıfların her birine ne kadar benzediğini ifade eden bir vektör olarak değerlendirilebilir.

Ağ çıktısı, ilk konvolüsyon katmanı da dahil olmak üzere sonraki herhangi bir katmandan alınabilir. Her bir havuzlama katmanında sonra, ağdan geçmekte olan görüntünün boyutu yarı yarıya azalacağından, daha derin katmanlardan alınacak veri boyutu da daha düşük olacaktır. Ancak katmanlar ilerledikçe yakalanacak örüntülerin, eğitim aşamasında verilen görüntülerin sınıflarına daha özgü örüntüler olması beklenmektedir.

Bir görüntü daha önce eğitilmiş bir ağdan geçirildikten sonra ağın bir katmanından alınacak çıktı, görüntünün sınıfı için direkt olarak anlam ifade etmeyecektir. Alınan değerlerin görüntü için ayırt edici olması bekleneceğinden, bu değerler öznitelik olarak değerlendirilip diğer bir aşamada başka bir sınıflandırıcı yardımıyla, görüntüler hakkında istenen değerlendirmenin yapılmasını sağlayacaktır (Şekil 2.11).



Şekil 2.11. Önceden eğitilmiş bir konvolüsyonel yapay sinir ağının kullanımı

Böyle bir amaç ile kullanılacak bir konvolüsyonel yapay sinir ağının, mümkün olduğu kadar fazla örüntü öğrenebilmesi için, ağın eğitildiği görüntülerin çeşitliliği ve sayısı da önemlidir. Ağ ne kadar çok fazla sınıf ve görüntü ile eğitilirse, ağın öğrenebileceği o kadar çok örüntü ve yeni görüntüleri benzetebileceği o kadar fazla sınıf olacaktır.

2.4. Sınıflandırma Yöntemleri

Tez kapsamında yapılan çalışmalarda elde edilen özniteliklerin sınıflandırılmasından, k-en yakın komşu yöntemi ve destek vektör makineleri kullanılmıştır.

k-en yakın komşu yöntemi, test edilen örneği temsil eden vektör ile eğitim setindeki örnekleri temsil eden vektörler arasındaki uzaklıkları dikkate alarak sınıflandırma yapan bir sınıflandırıcıdır. İki vektör arasındaki uzaklığın hesaplanmasında en çok kullanılan uzaklık ölçüm metotlarından bir tanesi Öklid uzaklığıdır (Eşitlik 2.14). x_i ve x_j eğitim ve test veri setlerinde yer alan iki ayrı vektörü temsil etmektedir.

$$d(x_i, x_j) = \sqrt{\sum_k (x_{ik} - x_{jk})^2} \quad (2.14)$$

Destek vektör makineleri ayrı sınıflara ait vektörler arasına yerleştirilen bir hiper düzlemin sınıflara uzaklığının optimize edilmesine dayalı bir sınıflandırma metodudur. CNMFET içerisinde metodun ayrıca implementasyonu yapılmamış ve destek vektör makineleri ile sınıflandırma yapılabilmesi için LIBSVM (Chang ve Lin, 2011) kullanılmıştır.

2.5. Performans Ölçümü

Sınıflandırma sonuçları, genelde karmaşıklık matrisi üzerinden elde edilen bazı ölçütlerle değerlendirilir. Karmaşıklık matrisi, her bir sınıfa ait tüm örneklerin, sınıflandırıcı tarafından hangi sınıfa ait olarak kaçar defa tespit edildiğini gösteren bir matristir (Çizelge 2.2). Bu matris üzerinden sınıflar ve sınıflandırma başarısıyla ilgili, gerekli pek çok çıkarım yapılabilir;

- Her bir satırdaki sayıların satır içi toplamı, o satırın sınıfındaki toplam örnek sayısını verir.
- Köşegende yer alan her bir sayı, bulunduğu satırın sınıfında kaç örneğin doğru sınıflandırıldığını gösterir.
- Köşegendeki sayıların toplamı, sınıflandırma sonucunda doğru sınıflandırılan toplam örnek sayısını verir.
- Matristeki tüm sayıların toplamı, sınıflandırılan toplam örnek sayısına eşit olur.

Çizelge 2.2. Örnek bir karmaşıklık matrisi

a	b	<--	sınıflandırma sonucu
k	m		a = Sınıf1
n	p		b = Sınıf2

Aynı matris örneği üzerinden devam edildiğinde;

- k sayısı, a sınıfında olup doğru sınıflandırılan örnek sayısını,
- m sayısı, a sınıfında olup b sınıfında yanlış sınıflandırılan örnek sayısını,
- n sayısı, b sınıfında olup a sınıfında yanlış sınıflandırılan örnek sayısını,
- p ise, b sınıfında olup b sınıfında doğru sınıflandırılan örnek sayısını verir.

Tez çalışması kapsamında yapılan tüm testlerde, sınıflandırma sonuçları, doğru sınıflandırılan örnek sayısının, tüm örnek sayısına oranı 100 ile çarpılarak elde edilen doğruluk değeri cinsinden verilmiştir. Çalışmaların birinde karşılaştırma yapılabilmesi amacıyla kesinlik, duyarlılık ve F1 skorları da verilmiştir.

Bir sınıfa ait olan ve doğru sınıflandırılan örnekler DP (doğru pozitif), sınıfa ait olan ve yanlış sınıflandırılan örnekler YP (yanlış pozitif), sınıfa ait olmayan ve doğru sınıflandırılan örnekler DN (doğru negatif), sınıfa ait olmayan ve yanlış sınıflandırılan örnekler YN (yanlış negatif) ile temsil edildiğinde, kesinlik, duyarlılık ve F1 skorları sırasıyla eşitlik 2.15, 2.16 ve 2.17 olarak hesaplanır.

$$kesinlik = \frac{DP}{DP+YP} \quad (2.15)$$

$$duyarlılık = \frac{DP}{DP+YN} \quad (2.16)$$

$$F1 \text{ skoru} = \frac{2}{\frac{1}{kesinlik} + \frac{1}{duyarlılık}} \quad (2.17)$$

2.5.1. k-parçalı çapraz doğrulama

Çapraz doğrulama, yapılan bir sınıflandırma veya analiz işleminin, veri seti üzerinde nasıl sonuçlanacağını ölçmek için kullanılan bir doğrulama metodudur. Bir veri setindeki örneklerin tek seferde eğitim ve test olmak üzere iki parçaya bölünmesi ve çalışmaların bu iki alt set üzerinde yapılması da mümkündür. Fakat böyle bir durumda eğitim ve test, veri setindeki her bir örnek için uygulanmış olmayacaktır.

Çapraz doğrulama tekniğinde veri seti, k parçaya bölünür. Bu k parçadan $k-1$ tanesi eğitim için kullanılırken, kalan 1 parça testte kullanılır. Bu işlem, her bir parça tek başına testte kullanılana kadar k defa tekrar edilir. Böylece her bir örnek, ayrı ayrı hem eğitim hem de test işleminde yer alır. Ayrıca hiçbir adımda testte yer alan bir görüntünün eğitimde yer almasına izin verilmemiş olur.

k değeri veri setinin büyüklüğüne bağlı olarak farklı değerlerde seçilebilir. Genelde $k=10$ olarak seçilir. Seçilen değer, veri setinde en az örneğe sahip sınıfın örnek sayısına eşit veya bu sayıdan büyük olması önem taşır. Eğer daha küçük olursa, bu sınıf için hiçbir adımda test setinde örnek bulunmayacaktır.

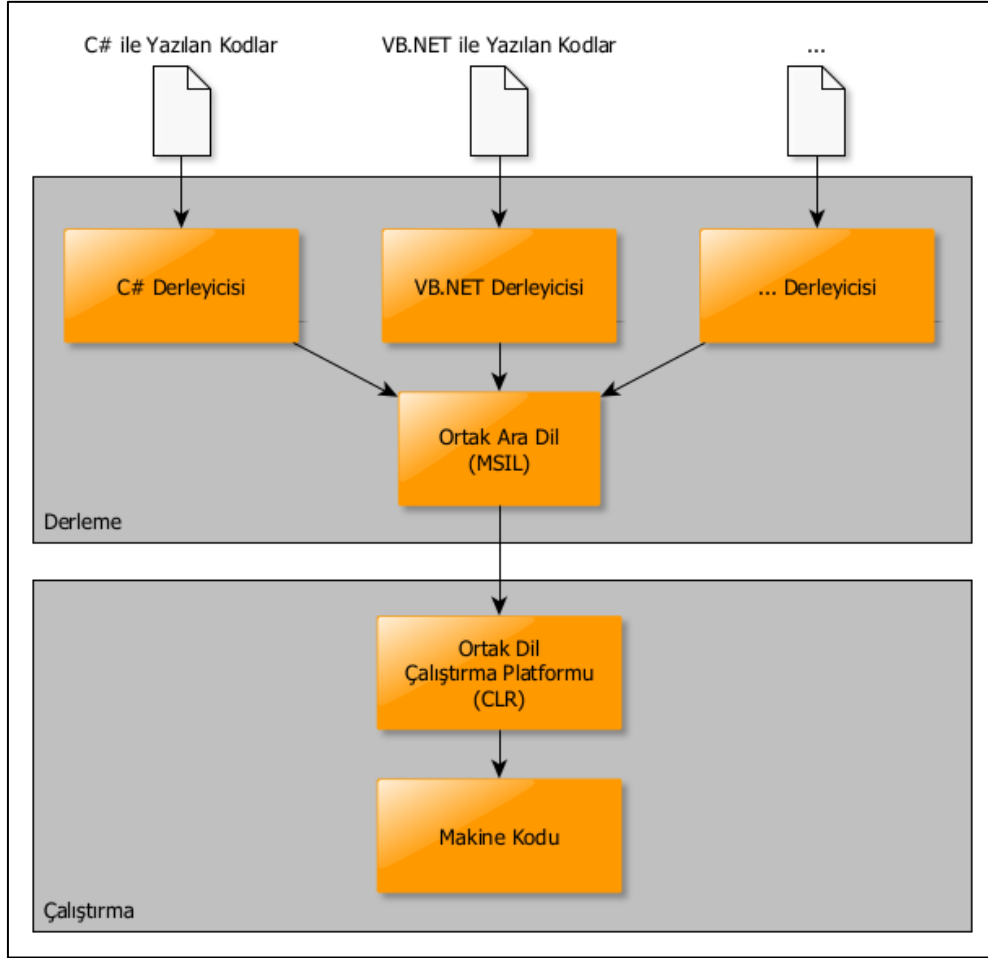
Yöntem pratiğe dökülürken örneklerin rastgele seçilmesi de oldukça önemlidir. Örnekler parçalara rastgele dağılmalı, eğitimde yer alan bir veri testte, testte yer alan veri eğitimde kesinlikle yer almamalıdır.

Tez çalışması boyunca son çalışma haricinde yapılan tüm testlerde, $k=10$ parçalı çapraz doğrulama yapılmıştır. Son çalışmada ise k değerinin, sonuçların tutarlılığına fazlaca etki etmediğinin gösterilebilmesi için $k=10$ ile birlikte, $k=5$ ve $k=2$ de test edilmiştir.

2.6. Yazılım Geliştirme Ortamı

Tez kapsamında geliştirilen kütüphane ve araç seti, C# ile, Visual Studio 2022 kullanılarak, .NET çatısı altında yazılmıştır. Visual Studio, farklı diller ile çeşitli platformlar için yazılım hazırlanabilmesini sağlayan ve ücretsiz sürümleri de bulunan, Microsoft tarafından sürülen ve desteklenen gelişmiş bir geliştirme ortamıdır. .NET ise çok sayıda kütüphanesi bulunan bir çatıdır.

.NET çatısı, altında pek çok farklı dilde geliştirme yapılabilmesine imkan tanıyan bir yapıda tasarlanmıştır. Desteklenen farklı dillerde yazılan kodlar, direkt makine koduna dönüştürülerek çalıştırılabilir dosyalara derlenmez. Kodlar, önce bir ortak ara dile derlenir (CIL, Common Intermediate Language). Microsoft tarafından geliştirilen bir ortak ara dil olan MSIL (Microsoft Intermediate Language) diline derlenen kodlar çalışma anında bir ortak dil çalışma platformu olan CLR (Common Language Runtime) tarafından çalışma anında makine kodlarına çevrilip çalıştırılır (Şekil 2.12).



Şekil 2.12. Kodların .NET çatısı altında ortak dile derlenmesi ve çalıştırılması

.NET çatısı altında bellek alanları CLR tarafından yönetilir. Bellek alanlarının CLR tarafından yönetildiği kodlar, yönetilen kodlar olarak isimlendirilir. Ancak bellek alanları üzerinde daha fazla kontrol gerektiğinde, CLR bellek yönetimini yazılan yazılıma bırakabilir. Bellek ayırma ve serbest bırakma işlemlerini kendi kontrolü altında tutan kodlar, düzgün yazılmadıklarında bellek yönetimi açısından güvenli olmayabileceği için bu kodlara güvenli olmayan kodlar, yazıldıkları bağlama güvenli olmayan sınıf veya güvenli olmayan fonksiyon olarak isimlendirilir. Bellek yönetiminin CLR'a bırakıldığı kodlar ise belleğe direkt erişime sahip olmadıkları için güvenli kodlar olarak adlandırılır.

2.6.1. Paralel kod yürütme desteği

Günümüzde işlemcilerin büyük çoğunluğu artık tek bir çekirdek yerine, aynı anda işlem yürütebilen birden fazla çekirdek ile üretilmektedir. Bununla birlikte, yazılımlar aynı anda paralel işlem yürütme kabiliyetiyle yazılabilmekte ve normalde tek bir koldan ilerlendiğinde yapılabilecek işlemler, işlemci çekirdeklerinin sayısı ile doğru orantılı bir şekilde göreceli

olarak çok daha hızlı yapılabilir. Ancak paralelleştirme, aynı bellek alanlarının paylaşımı, girdi veya çıktılarının birbirlerini bağımlı olması gibi bazı sorunları da beraberinde getirmektedir.

.NET çatısı, iş parçacıkları oluşturmak ve yönetmek için gerekli desteği sunmanın yanında paralel döngüleri de desteklemektedir.

2.6.2. Karşılıklı dışlayıcılar

Bir işletim sistemi üzerinde çalışan bir veya birden fazla işleme ait iş parçacıkları ortak bir kaynağa erişim gerektirebilir. İşletim sistemi, işlem gücünü iş parçacıklarına oldukça kısa süreler için döngüsel olarak ayırırken, bir iş parçacığının ortak bir kaynak üzerinde yaptığı bir işlemin, sıra başka bir iş parçacığına geldiğinde yarım kalması olasıdır. İşlem sırası tekrar aynı iş parçacığına geldiğinde, ortak kullanılan kaynağın başka bir iş parçacığı tarafından değiştirilmiş olması mümkündür. Böyle durumlarda bir iş parçacığının, örneğin bellek gibi bir ortak alana başka iş parçacıklarının erişimini geçici olarak duraklatmasını mümkün kılmak için mutex (mutual exclusion / karşılıklı dışlama) yapıları kullanılır.

Mutexler basit bir bayrak değişkeni gibi çalışır. Ancak bir mutex üzerinde yapılan işlem yarıda kesilmez. Böylece bir iş parçacığı akış esnasında belli bir kısmı kilitleyerek başka bir iş parçacığının o kısma girişini, kilit aynı iş parçacığı tarafından açılana kadar bekletebilir.

Mutexler, ortak ve yoğun bellek kullanımı gerektiren uygulamalarda bellek yönetimi için oldukça kullanışlı yapılardır.

2.6.3. Temel doğrusal cebir altprogramları (BLAS) kütüphaneleri

Temel doğrusal cebir altprogramları (Basic Linear Algebra Subprograms, BLAS) olarak isimlendirilen fonksiyonlar, matris çarpımı, nokta çarpım, doğrusal kombinasyon gibi bazı doğrusal cebir işlemlerini yürütmek üzere yazılan standartlaşmış düşük seviyeli fonksiyonlardır. İşlemci mimarilerine işlemci geliştiricileri tarafından optimize edilen bu fonksiyonlar işlemcilerin sahip olduğu hızlandırma teknolojilerini de kullanarak çalıştıklarından, tez çalışması kapsamında ihtiyaç duyulan matris çarpımı gibi işlemlerin klasik metotlara göre çok hızlı bir şekilde yapılabilmesini sağlar.

Tez çalışması kapsamında hazırlanan yazılım, BLAS desteğine ihtiyaç duymadan çalışabilecek şekilde yazılmış, ancak açılışta BLAS desteğinin varlığını tespit ederek, bulunması haline iki farklı BLAS kütüphanesini de kullanabilecek şekilde tasarlanmıştır. Bunlardan bir tanesi Intel MKL (Math Kernel Library), diğeri açık kaynaklı olarak geliştirilip dağıtılan ve AMD işlemciler için de destek sağlayan openBLAS kütüphanesidir.

Kütüphanelerden herhangi birinin varlığı halinde, konvolüsyon için uygulanan matris çarpımları, tek duyarlılıklı kayan noktalı sayı tipi üzerinde matris çarpımı yapan sGeMM fonksiyonuna yaptırılarak hızlandırılmaktadır.

2.6.4. Düzenli ifadeler

Düzenli ifadeler (regex – regular expression), yazılım dillerinde metinsel ifadelerden belli kısımların tespit edilebilmesine, ayıklanabilmesine imkân sağlayan bir çeşit ifade dilidir. Desenler şeklinde tanımlanan düzenli ifadeler sayesinde, metinler içerisinde desenlerle eşleşen kısımların tespiti kolaylaşır.

Tez çalışması kapsamında yazılan yazılımın arayüzünde, araştırmacıların veri seti olarak kullanacakları görüntü dosyalarının isimleri ve yolları üzerinden sınıf isimlerinin belirlenebilmesi amacıyla düzenli ifadeler kullanılmıştır.

3. İMPLEMENTASYON

Tez çalışması kapsamında konvolüsyonel yapay sinir ağlarının ileri besleme aşamasının ve temel katmanlarının implementasyonları yapılarak CeNiN isminde bir kütüphane yazılmıştır. Yazılan kütüphane hem açık kaynak kodlu olarak yayınlanmış, hem de yazılan kullanıcı arayüzü içerisinde CeNiN isim uzayı altında gömülmüştür. Kullanıcı arayüzü ile birlikte öznitelik çıkarımı, öznitelik seçimi, sınıflandırma gibi işlemlerin kolayca yapılabilmesini sağlamak amacıyla da CNNFET olarak isimlendirilen bir araç seti geliştirilmiş ve açık kaynak kodlu olarak kullanıma sunulmuştur.

3.1. Konvolüsyonel Yapay Sinir Ağı Kütüphanesi; CeNiN

CeNiN, temel katmanların ve ileri besleme aşamalarının implementasyonlarını içeren bir konvolüsyonel yapay sinir ağı kütüphanesi olarak geliştirilmiştir. Yeni katmanlar için kolaylıkla genişletilebilir yapıda hazırlanmıştır. Katmanları temsil eden sınıflar ile birlikte diğer sınıfların diyagramları EK A ve EK B’de sunulmuştur.

Bir konvolüsyonel yapay sinir ağının ileri besleme aşaması bellekte çok fazla veri ve işlem gerektirir. Bunun için kütüphaneye yazılan ilk sınıflardan biri *Tensor* sınıfıdır. Bellekte gerekli alanı hesaplayıp bellek alanı ayıracak fonksiyonlar buraya yazılmıştır. Sınıfta, *Tensor* tipinde iki nesnenin farkı, toplamı ve çarpımı için gerekli işlemlerle birlikte indeksleme ve yeniden boyutlandırma işlemleri için ihtiyaç duyulan fonksiyonlar da eklenmiştir. Bellek üzerinde işaretçilerle işlem yapmak ve bellek yönetimini .NET’in CLR’ına bırakmamak için *Tensor* sınıfı, tezin 2. bölümünde amacı ve anlamı özetlenen “güvenli olmayan” sınıf olarak tanımlanmıştır. Bellek üzerinde ayırma, serbest bırakma ve dolaşım işlemlerini yapan fonksiyonlar da “güvenli olmayan” fonksiyonlar olarak yazılmıştır. Çok sık oluşturulup yıkılacak olan *Tensor* sınıflarından türeyen nesnelerin yıkıcılarında, ayrılan bellek alanlarının serbest bırakılması ve temizlik işlemleri yapılmıştır.

Layer katmanı bir özet sınıf olarak tasarlanmıştır. Bu katmanın üyelerini miras alan diğer katmanlarda kullanılmak üzere, ihtiyaç duyulan ortak özellik, alan ve metotlar bu sınıfta yazılmıştır.

Katmanlar işleyecekleri ve işledikleri verileri tutacak iki farklı alana ihtiyaç duyar. Ancak katmanları temsil eden *Layer* sınıfına yalnızca giriş verileri için bir *Tensor* nesnesi yerleştirilmiştir. Her katman işlediği verileri bir sonraki katmanın girişine yazacak şekilde tasarladığından, *Layer* sınıfına ikinci bir *Tensor* nesnesi yerleştirilmemiştir. Böylece bir katmanın çıkış verilerinin sonraki katmanın girişine kopyalanmasına gerek yoktur.

Ağın dosyadan yüklenmesi, içerisinde yer alan katmanların oluşturulması, *CNN* sınıfında yapılmaktadır. Sınıf içerisinde giriş katmanı, çıkış katmanı ve tüm katmanların bir dizisi tutulur. Katmanlar *Layer* üst sınıfında uç uca bağlanıp (*appendNextLayer*) bir zincir meydana getirecek şekilde tasarlandığından, sadece giriş katmanı üzerinden, her katmanda bir sonraki katmana geçilerek (*nextLayer*) çıkış katmanına erişilebilir.

Konvolüsyonel yapay sinir ağları milyonlarca değişken içerebildiğinden, katman yapılarının ve bu değişkenlerin çok hızlı bir biçimde belleğe yüklenebilmesi gerekir. Bu amaçla *CNN* sınıfı içerisinde belleğe yüklenen ve Çizelge 3.1’te özetlenen .cenin uzantılı bir dosya formatı tasarlanmıştır.

Çizelge 3.1. “.cenin” dosya formatı

Veri	Açıklama
“CeNiN NEURAL NETWORK FILE”	string (sonlandırma karakteri olmadan)
Katman sayısı	1 x int32
Giriş boyutu	3 x int32
Ortalama piksel	3 x float
Her bir katman için	
Katman tipi metin uzunluğu	1 x byte (7-bitlik tamsayı)
Katman tipi	string
Katman parametreleri	Katmanlara özel parametreler (padding ve stride değerleri, konvolüsyon filtreleri vs.)
“EOF”	string (sonlandırma karakteri olmadan)

Her katmanla paylaşılacak şekilde *Layer* sınıfına birer mutex yerleştirilmiştir. Burada amaç, paralelleştirme yapıldığında bir iş parçası bir katmanın giriş tensöründeki verileri

işliyoruzken bu tensöre girişleri engellemektir. Bu konu Parallelleştirme başlığı altında detaylandırılacaktır.

Diğer katmanları temsilen yazılan ve *Layer* sınıfının üyelerini kalıtım yoluyla alan temel sınıflar Input, ReLU, Conv, Pool, SoftMax ve Output sınıflarıdır.

Input sınıfı görüntülerin dosyadan okunduğu sınıftır. Burada gerekiyorsa görüntüler konvolüsyonel ağı giriş katmanının boyutlarına ölçeklenir.

ReLU sınıfında, katmanın giriş tensöründeki veriler aktivasyon fonksiyonu olan ReLU'dan geçirilip bir sonraki katmanın girişine yazılır.

Conv sınıfında, katmanın giriş tensöründeki veriler çok kanallı bir görüntüyü temsil eder. Bu görüntü, katmandaki filtrelerin tamamından bu katmanda geçirilir ve filtrelenmiş görüntü bir sonraki katmanın girişine yazılır.

Pool sınıfında, katmanın giriş tensöründeki görüntü havuzlama işleminden geçirilip bir sonraki katmanın girişine yazılır.

Softmax katmanı, aslında bir önceki katmanda elde edilen değerlerin softmax fonksiyonu ile olasılık temsil eden değerlere dönüştürüldüğü son katmandır. Ancak başta *Layer* sınıfı, bu sınıfta ve dolayısıyla onun üyelerini miras alan diğer hiçbir sınıfta bir çıkış belleği bulunmayacak şekilde tasarlandığından, *Output* isimli bir sınıf daha eklenmiştir.

Output sınıfı, bir önceki katmanda artık ağda ileri besleme aşamasını tamamlayan görüntü için sonuçların tutulduğu, olasılıkların ve karşılık gelen kategori/sınıf isimlerinin de yer alacağı bir sınıf olarak tasarlanmıştır.

CBLAS sınıfı içerisine, kütüphaneye CBLAS desteği kazandırmak için gerekli fonksiyonlar yazılmıştır. Sınıf içerisinde, CBLAS için gerekli kütüphanelerin çalışılan sistemdeki mevcudiyetinin kontrolü, işlemci çekirdek sayısına bağlı olarak kaç iş parçacığı kullanılabileceğinin belirlenmesi ve CBLAS desteğiyle matris çarpımı için gerekli fonksiyon çağrılarının tanımları yazılmıştır. *CBLAS*, Intel MKL ve openBLAS kütüphanelerini kullanabilmektedir. BLAS desteğinin hıza katkısı Parallelleştirme başlığı altında detaylandırılacaktır.

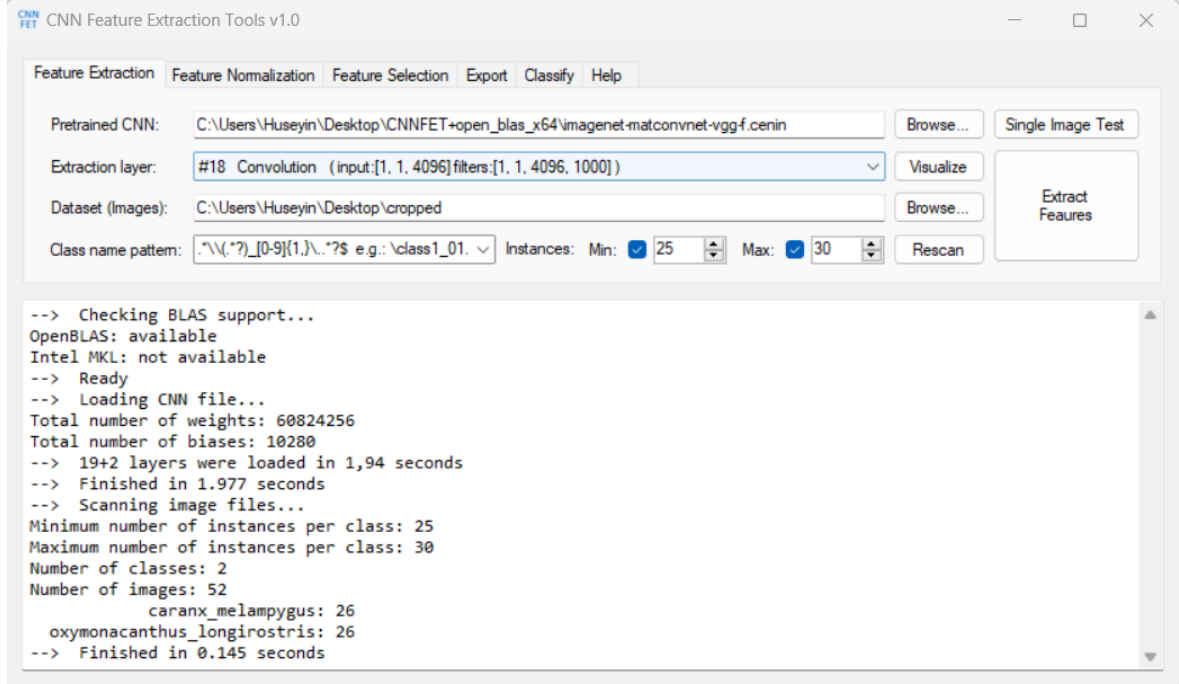
3.2. Konvolüsyonel Yapay Sinir Ağları Öznelik Çıkarım Araçları (CNNFET)

Tez çalışması kapsamında, CNNFET (Convolutional Neural Networks Feature Extraction Tools / Konvolüsyonel Yapay Sinir Ağları Öznelik Ayıklama Araçları) olarak isimlendirilen bir araç seti hazırlanmıştır. Hazırlanan yazılım, tez kapsamında Microsoft .NET çatısı altında yazılmış olan konvolüsyonel yapay sinir ağı kütüphanesi CeNiN'den ve arayüzleri de içeren Windows form uygulamasından oluşmaktadır.

CNNFET, her aşama için bir sekme içeren bir arayüz ile başlar (Şekil 3.1). Sekmelerin altında, yürütülen işlemlerle alakalı bilgilerin aktığı bir alan mevcuttur. Bu alana klavyede “CTRL” tuşuna basılı iken çift tıklandığında alandaki bilgiler temizlenebilir. Uygulamanın arayüzleri alt başlıklar şeklinde detaylandırılmıştır.

3.2.1. Öznelik çıkarımı

Aynı zamanda ana ekran olan bu sekmede (Şekil 3.1), ön eğitilmiş konvolüsyonel yapay sinir ağının seçimi ile başlanır. Sonrasında ağ belleğe yüklenir ve içerdiği kamanlar listelenir. Kullanıcı bu aşamada “Single Image Test” butonu ile ağdan bir görüntü geçirip, ağın görüntüde ne “gördüğünü” test edebilmektedir. Sonrasında öznelik ayıklama işleminin yapılacağı katman seçilir. Seçilen katmandaki konvolüsyon filtreleri “Visualize” butonuna tıklanarak görsel olarak listelenebilir (Şekil 3.2). Bir sonraki aşamada veri setini meydana getiren görüntü dosyalarının yer aldığı dizinin seçimi yapılır. Uygulama, tespit edilen sınıfların ve her sınıfta kaç dosya yer aldığının bilgilerini verecektir. “Class name pattern” alanından başka bir ön tanımlı ifade seçilerek veya farklı bir ifade girilerek sınıfların tekrar taranması sağlanabilir. Aynı ekran üzerinde bir sınıfın sınıf olarak kabul edilebilmesi için en az kaç örnek içermesi gerektiği veya her sınıftan en fazla kaç örnek alınacağı sınırlandırılabilir. Seçimler tamamlandığında, “Extract Features” butonu ile öznelik çıkarım işlemi başlatılır.



Şekil 3.1. CNNFET araç seti

Arayüzde “Class name pattern” kısmına, iki ön tanımlı düzenli ifade girilmiştir.

.*\\(.*)\\.\\.*?\\.*?\$ şeklinde girilen ilk ön tanımlı ifadede;

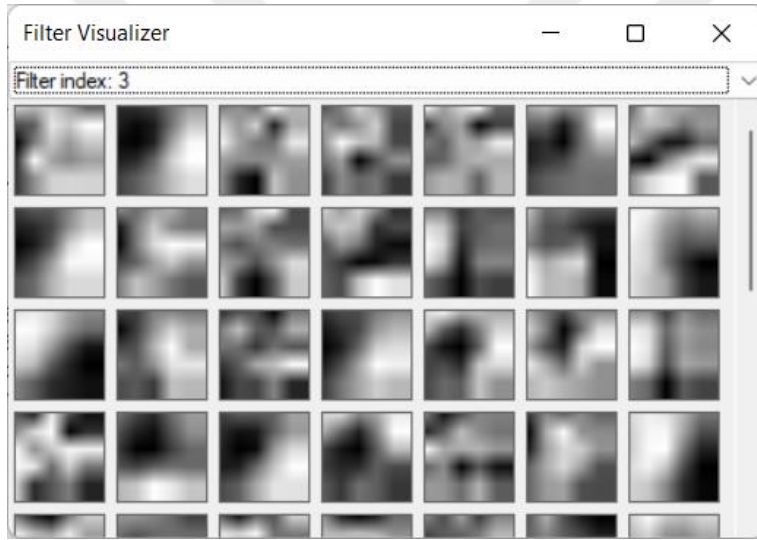
- “.” herhangi bir karakter ile eşleşir.
- “*” herhangi karakterden 0 veya fazla defa bulunabileceğini belirtir.
- “\\” ifadesinde tek “\” karakteri C# kodu içerisinde çift tırnak içinde bir metinde yer aldığından derleyiciden kaçınılıp düzenli ifade yorumlayıcısına tek “\” olarak gönderilebilmesi için başına bir “\” karakteri daha eklenerek yazılmıştır.
- “(.*)” kısmı, sınıf adını temsilen yakalanmak istenen kısım olduğu için parantez içine alınmıştır. Yukarıda belirtildiği gibi herhangi bir karakterden 0 veya daha fazla bulunabileceğini ifade eder. “?” karakteri, desene uygun, mümkün olan en az sayıda karakter eşleşmesinin kabul edilmesini, böylece daha sonra gelmesi istenen “\” karakterinin de bu eşleşmeye dahil olmamasını sağlar.
- Devamında gelen “\\.\\.*?\\.*?” ifadesi klasörün içindeki dosyanın adını ve uzantısını ifade eder (Örneğin \dosyaadı.uzantı). “.” karakteri, düzenli ifadelerde “herhangi bir karakter” anlamına geldiği için “\” ile yorumlayıcıdan kaçırılmıştır.
- Sondaki “\$” karakteri, metin içerisinde ortalarda bir parça çekilmeyeceğini, metnin bu şekilde sonlanması gerektiğini ifade etmektedir.

Böylece örneğin “C:\Dosyalar\Veri Seti\sınıfl\goruntu1.jpg” gibi bir dosya yolunda, “sınıfl” metni goruntu1 dosyasının temsil ettiği sınıf olarak yakalanacaktır.

.*\((.*?)\)[0-9]{1,}\.?.*?\$ şablonuna göre ise yukarıda olmayıp açıklanması gereken ifadelere bakıldığında;

- “[0-9]{1,}” en az bir tane 0-9 arasında rakam geleceğini ifade eder.

Böylece örneğin “C:\Dosyalar\Veri Seti\sınıfl_001.jpg” gibi bir dosya yolunda, “sınıfl” metni, sınıfl_001 dosyasının temsil ettiği sınıf olarak yakalanacaktır.



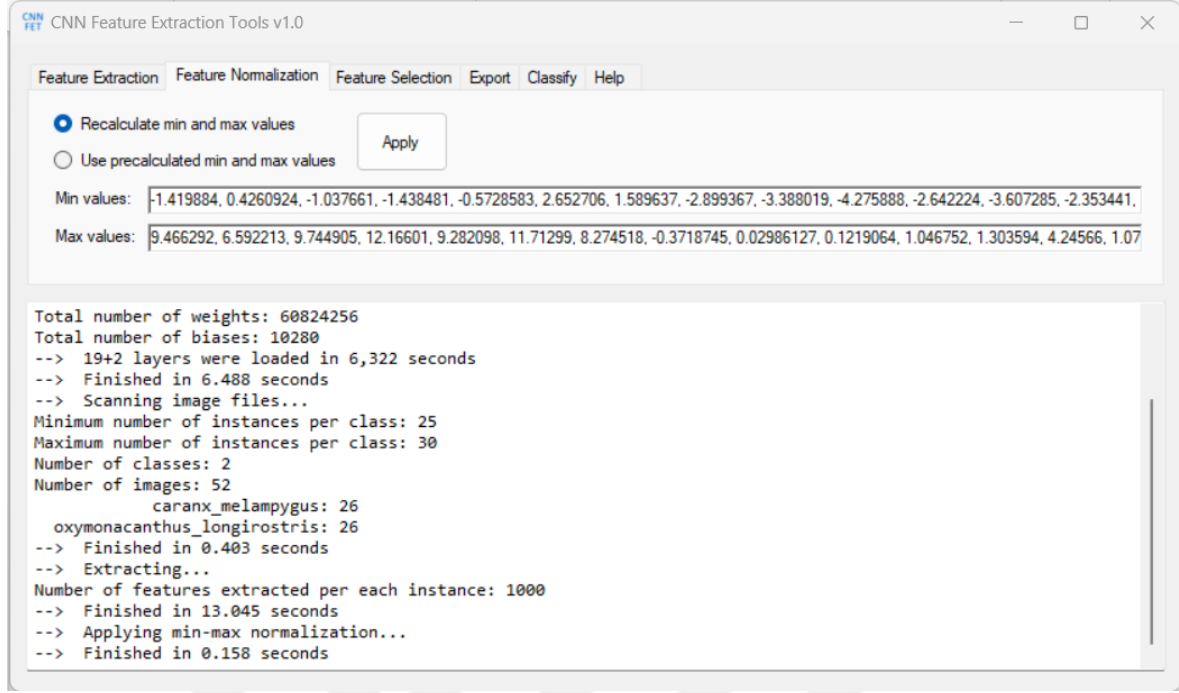
Şekil 3.2. Konvolüsyon filtrelerinin görselleştirilmesi

3.2.2. Öznitelik normalizasyonu

CNNFET’in öznitelik normalizasyonu sekmesi (Şekil 3.3), çıkarımı yapılan özniteliklerin 0-1 aralığına normalize edilebilmesi sağlar (Eşitlik 3.1).

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3.1)$$

Normalizasyondan sonra her bir öznitelik vektöründe hesaplanan minimum ve maksimum değerler de kopyalanabilir şekilde arayüze aktarılmaktadır. Böylece daha sonra, örneğin bir test veri seti üzerinde, eğitim verileri üzerinde elde edilmiş minimum ve maksimum değerlerle normalizasyon yapılabilmesi mümkün olmaktadır.



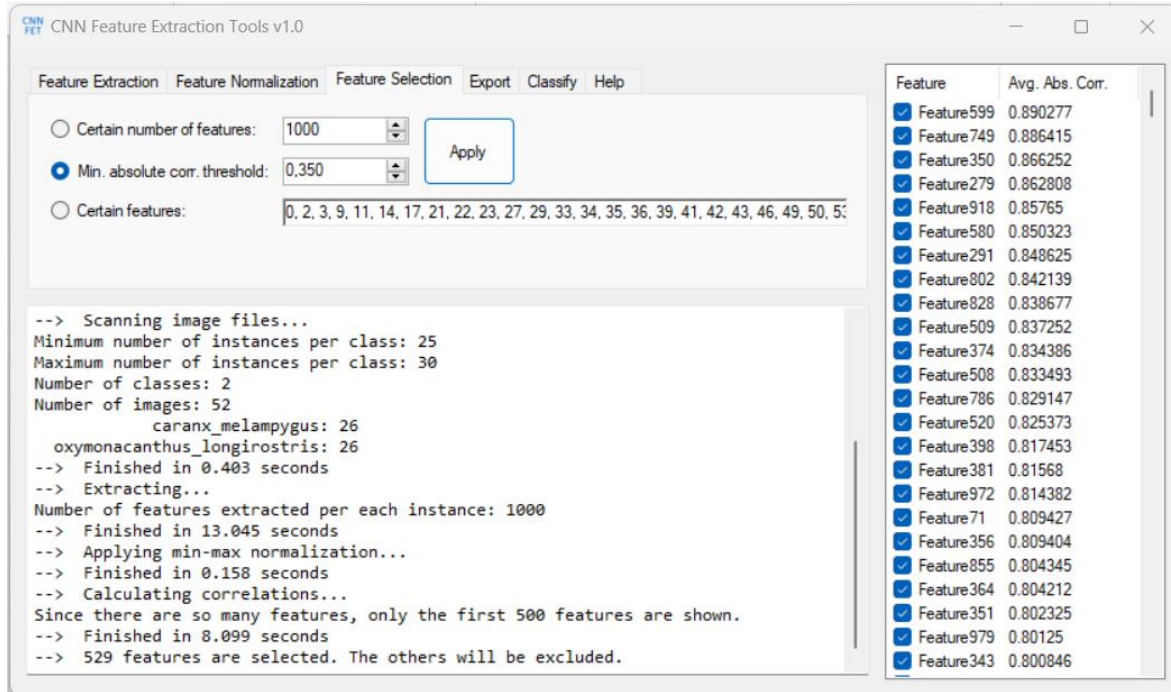
Şekil 3.3. Öznitelik normalizasyonu

3.2.3. Öznitelik seçimi

Öznitelik seçimi sekmesi tıklandığında (Şekil 3.4), çıkarımı yapılan öznitelikler ile sınıflar arasındaki Pearson korelasyonlarının (Bobadilla, Fernando, Antonio, 2012) (Eşitlik 3.2) mutlak değerleri hesaplanıp büyükten küçüğe doğru sıralanır. Kullanıcıya öznitelikleri korelasyon değerlerine göre filtrelemek için üç seçenek sunulur:

- Birinci seçenek seçilirse, en yüksek korelasyona sahip belli sayıda öznitelik seçilerek diğerleri sonraki aşamalarda yok sayılır.
- İkinci seçenekte bir korelasyon eşiği yazılarak bu eşiğin altında kalan özniteliklerin elenmesi sağlanır.
- Bir diğer seçenekte ise hangi özniteliklerin seçileceği, öznitelik numaraları elle girilerek (ilk özniteliğin numarası sıfırdır) belirlenir.

$$r_{xy} = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}} \quad (3.2)$$

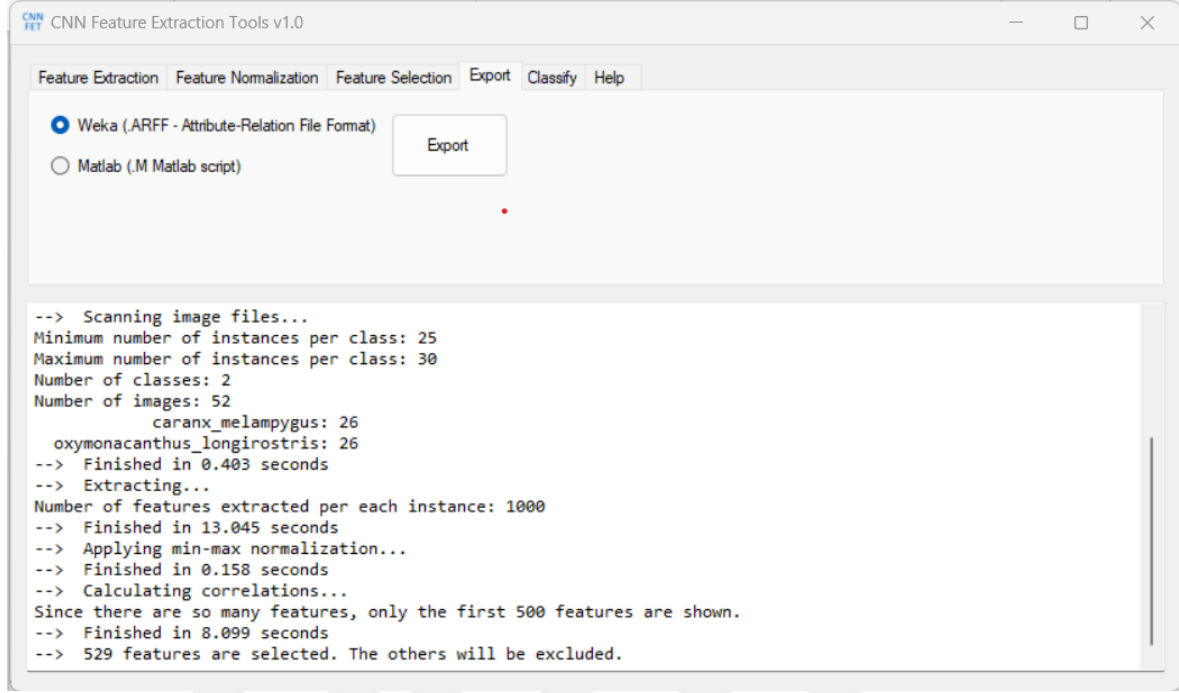


Şekil 3.4. Öznitelik seçimi

3.2.4. Farklı dosya formatlarına aktarım

CNNFET ile çıkarımı yapılan ve isteniyorsa normalizasyon ve öznitelik seçim işlemlerine tabi tutulan öznitelikler, “Export” sekmesi kullanılarak farklı formatlarda dışarı aktarılabilir (Şekil 3.5).

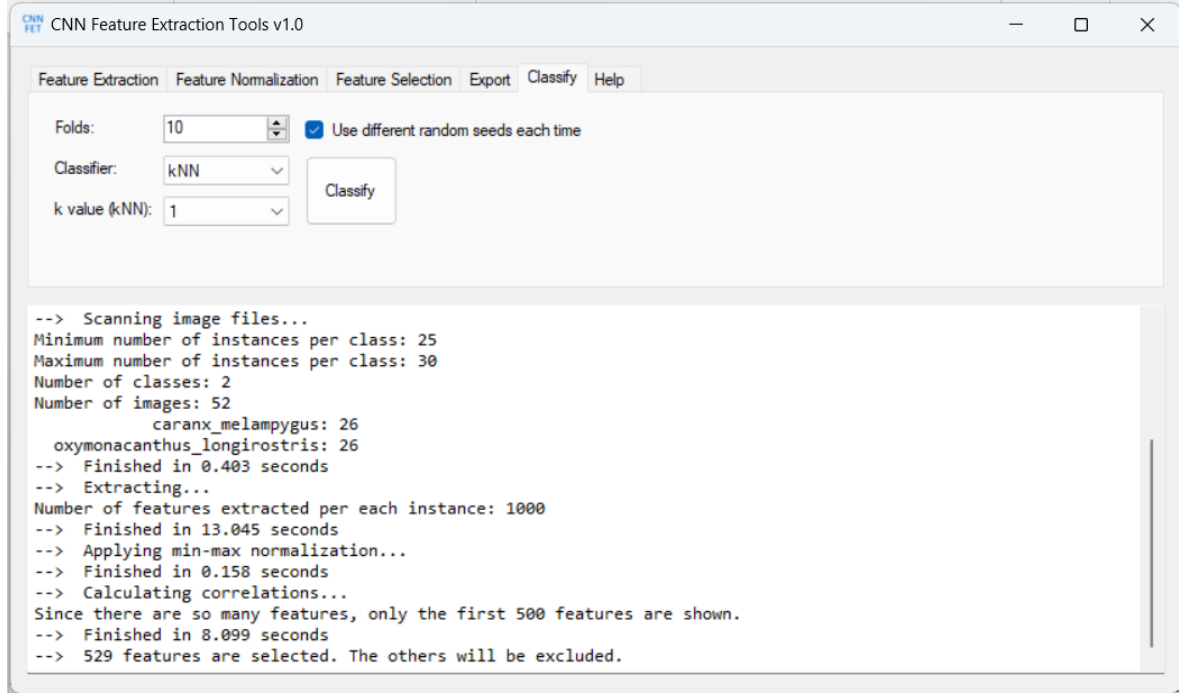
İlk format Weka (Frank, Hall, Ian, 2016) aracıyla uyumlu .arff formatında çıktı alınmasını sağlar. Diğer seçenek ise özniteliklerin ve sınıfların Matlab ortamına aktarılabilmesini sağlayan .m formatında bir betik üretir.



Şekil 3.5. Farklı dosya formatlarına veri aktarımı

3.2.5. Sınıflandırma

Classify sekmesi (Şekil 3.6) kullanıcıya, uygulama üzerinde sınıflandırma yapabilme imkânı verir. Sınıflandırma işlemleri k-parça çapraz doğrulama metodu ile yapılır. Çapraz doğrulama için hazırlanacak olan k adet parçada yer alacak örneklerin sıralaması bir rastgele üretici ile karıştırılmaktadır. Rastgele sıra üretici, testlerin tekrarlanabilirliği için uygulama her açıldığında aynı rastgele sıralamayı verecek şekilde hazırlanmıştır. Eğer rastgele sıralama da her seferinde farklılaştırılmak istenirse “Use different random seeds each time” seçeneği işaretlenmelidir.



Şekil 3.6. CNNFET ile sınıflandırma

3.3. Arayüz Uygulaması İçerisindeki Sınıflar

Uygulama arayüzü içerisindeki sınıflar ve temel görevleri aşağıda listelenmiştir.

- *ARFFFileWriter, MFileWriter*: Veri setindeki her bir örnekten çıkarımı yapılan öznitelikleri .arff (Weka için) ve .m (Matlab için) formatlarında dosyaya yazma
- *FileScanner*: Yolu verilen dizin içerisinde, belirtilen uzantılara sahip dosyaları bir tarama
- *Dataset*: FileScanner nesnesi ile görüntüleri tarayıp, görüntülerin sınıflarını ve görüntülerle alakalı diğer bilgileri bellekte tutma
- *ConvOutputExtractor*: Verilen görüntüyü, belirlenen katmana kadar verilen yapay sinir ağının katmanlarından geçirip katmanın çıktısını alma
- *FilterVisualizer*: Seçilen konvolüsyon katmanında yer alan filtreleri görselleştiren arayüzün hazırlanması
- *KFoldProvider*: Verilen örnekleri, her bir örneğin bir defa sadece test setinden yer almasını garantileyerek rastgele olarak belirtilen sayıda parçaya ayırma
- *PearsonCorrelation*: Özniteliğin örneklerdeki değerleri ile sınıflar arasında ağırlıklandırılmış Pearson korelasyonunu hesaplama

- *KNNClassifier*: Örnekleri k-en yakın komşu algoritması sınıflandırma
- *frmMain*: Uygulamanın ara yüzünü ve temel akışını içerme

3.4. Dosya Formatları

CNNFET ile işlenecek veri setlerinden elde edilen veriler Matlab, GNU Octave ve Weka ile işlenebilecek dosya formatlarında dışarı aktarılabilir.

Matlab, içerisinde çeşitli araçlar barındıran popüler bir paket program olmanın yanında, barındırdığı yorumlayıcı ile birlikte çalışan yüksek seviyeli programlama dilinin de adıdır. Ticari olmasına karşın, sıkça kullanılan Matlab, verilerin işlenmesini kolaylaştıracak çok sayıda hazır araç ve komuta sahiptir.

GNU Octave, ticari olan Matlab'ın açık kaynak kodlu ücretsiz bir alternatifidir. Matlab betikleri, Octave içerisinde çalışan yorumlayıcı ile çoğunlukla uyumludur.

Geliştirilen araç setinin hazırladığı verilerin bu iki yazılımla işlenebilecek şekilde dışarı aktarılması mümkündür. Dışarı aktarılan dosya, sınıfların metin formatında yer aldığı hücre dizisi değişkeni tanımıyla başlar. Daha sonra her bir örneğe ait öznitelik değerleri ayrı satırlarda olacak şekilde özniteliklerin yer aldığı matris ile devam eder. Her satırın sonunda, satırdaki örneğin sınıfının sırasını temsil eden bir sayı yer alır ve son satırda buna dair bir açıklama bulunur (Şekil 3.7).

```

cnnfet76709_classes = { 'cls0', 'cls1', ... };
cnnfet76709 = [
-3.118562 3.216149 ... 0
-2.414646 2.197301 ... 0
...
...
...
];
%Each row represents one instance and ends with class index (starting from zero)

```

Şekil 3.7. Örnek bir .m çıktısı şablonu

Böylece dosya tıklanıp Matlab veya Octave ile açıldığında, ilgili değişkenler yüklenerek kullanıma hazır hale gelecektir.

Weka, Waikato Üniversitesi'nde geliştirilmiş, makine öğrenimi ile ilgili çok sayıda paketi olan açık kaynak kodlu bir yazılımdır (Frank, Hall, Ian, 2016). Pratik ve kullanışlı olması dolayısıyla oldukça popüler olan bu yazılım, işleyeceği verileri dışarıdan .arff formatında alabilmekte ve dışarıya aynı formatla aktarabilmektedir.

Çalışma kapsamında geliştirilen yazılım ile .arff (Attribute Relationship File Format/ Öznitelik İlişki Dosya Formatı) formatında çıktılar alınarak elde edilen verilerin, Weka ortamında işlenmeye devam edilebilmesi sağlanmıştır. Dışa aktarılan .arff dosyaları, öznitelik setinin isminin yer aldığı @relation ifadesi ile başlar. Daha sonra satır satır, sıralı olarak öznitelik isimleri ve veri tipleri @attribute ifadesi ile yazılır. Öznitelikler, ayıklandıkları görüntülerin sınıfları ile ilişkilendirileceğinden öznitelik tanımlarından sonra sınıf özniteliği eklenir. Sonrasında @data ifadesi altına, her bir satırda bir öznitelik vektörü ve metin formatında vektörün ait olduğu sınıf yer alacak şekilde öznitelikler dosyaya dökülür (Şekil 3.8).

```
@relation CNNFET38286

@attribute attr0 numeric
@attribute attr1 numeric
...
...
@attribute class {'cls0','cls1', ...}

@data
-3.118562 3.216149 ... 'cls0'
-2.414646 2.197301 ... 'cls0'
...
...
...
```

Şekil 3.8. Örnek bir .arff çıktısı şablonu

CNNFET'ten .arff formatında çıktı alındıktan sonra dosya, Weka ortamında, sunulan tüm diğer araçlarla işlenebilir halde açılacaktır.

3.5. Bellek Kullanımı, Performans İyileştirmeleri ve Parallelleştirme

Derin konvolüsyonel yapay sinir ağları, çok sayıda katman ve her konvolüsyon katmanında çok sayı filtre içerdiğinden, klasik sinir ağlarına kıyasla bellekte çok fazla alan kaplar. Bellek alanları üzerinde bu kadar çok veri ile işlem yapılırken, programlama dillerinin araya girerek

yaptığı hesaplamalar işlemlerin çok fazla zaman almasına neden olur. Bu nedenle tensörler ve matrisler üzerinde yapılması gereken işlemlerde bellek alanlarının kontrolü, altında çalışılan .NET çatısına bırakılmamış, *Tensor* isimli bir sınıf yazılarak işaretçiler yardımıyla, yazılan sınıf içerisinde yapılmıştır.

3.5.1. Bellek kullanımı

Tez çalışması kapsamında yazılan CeNiN kütüphanesinde, bellek ile ilgili işlemler *Tensor* sınıfı içerisinde yapılmaktadır. *Tensor* sınıfı, bir nesne oluşturulma anında bellek üzerinde ihtiyaç duyduğu miktarı hesaplar ve ardışık olacak şekilde kendine bu alanı ayırır.

Tasarlanan katman yapısı ile *Layer* sınıfında hem giriş hem de çıkış olarak kullanılan *Tensor* nesneleri, yalnızca katmanda görüntü işlenecekken bellekte gerekli alanı ayırır ve görüntü işlenip sonuçlar sonraki katmanın girişine yazıldığında, katmanın tensöründe giriş için ayırdığı bellek alanı serbest bırakılarak *Tensor* nesnesi bellekten temizlenir. Böylece konvolüsyonel yapay sinir ağı belleğe yüklendiğinde, kullanım esnasında ihtiyaç duyulacak bellek alanlarının sürekli olarak işgal edilmemesi sağlanmıştır.

Ayrılan alandan bir veri çekilmesi gerektiğinde, nesneye, eleman sayısı tensörün boyut sayısı kadar olan bir tamsayı dizisi gönderilir. Bu sayıların belirttiği konumun, fiziksel olarak tek boyutlu halde tutulan bellek alanında hangi noktaya denk geldiği hesaplanır ve sayı o alandan okunarak döndürülür.

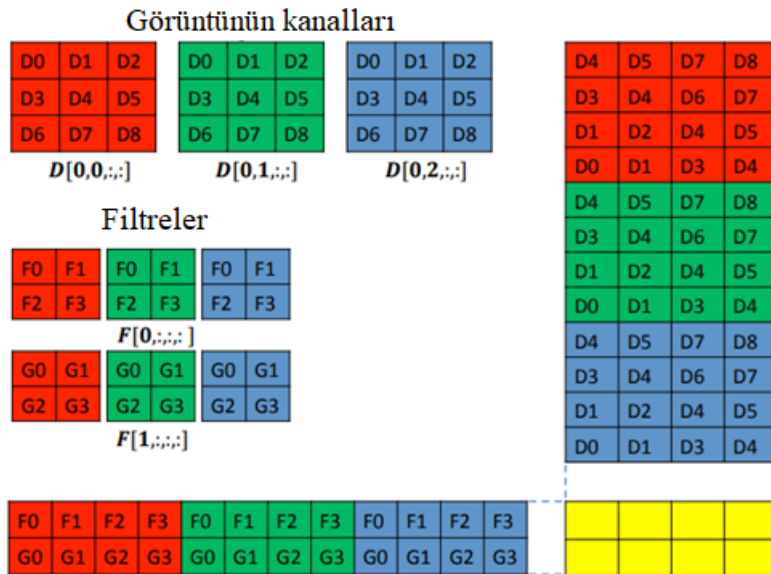
Çok boyutlu indislerden tek boyutlu indise geçiş bir taban dönüştürme işlemine benzer. Verilen indis dizisinde ilk elemandan sonraki her sayı, tensörün boyutlarının ardışık çarpımları ile çarpılır ve çarpımlar üzerlerine ilk eleman da eklenerek toplanır. Böylece okuma ve yazma anında indis dönüşümleri ve bellek alanları sınıfın kontrolünde tutulur.

3.5.2. Konvolüsyon işleminin hızlandırılması

Konvolüsyonel yapay sinir ağlarının ileri besleme aşamasında zaman açısından en maliyetli işlem konvolüsyon işlemidir. Konvolüsyon katmanı için yazılım tarafında dört ayrı implementasyon yapılmıştır.

İlk implementasyon (Conv_1) her bir filtre görüntü üzerinde gezdirilerek ve klasik iç içe döngüler kullanılarak yapılmıştır. Bu durum, bir görüntünün ağdan geçişinin oldukça yavaş olmasına sebep olmuştur.

İkinci implementasyonda (Conv_2), bir katmandaki çok sayıda filtre ve görüntü katmana varınca ayrıldığı her kanal ayrı ayrı gezilmek yerine katman içerisindeki tüm işlemlerin tek bir matris çarpımını indirgenmesini sağlayacak bir yöntem kullanılmıştır (Chetlur, Woolley, Vanderersch, Cohen, Tran, Catanzaro, Shelhamer, 2014). Çalışmanın ilerleyen bölümlerinde, işlemci tarafında tek komutla birden fazla veri ikilisi üzerinde işlem yapılabilmesini sağlayan SIMD (Single Instruction Multiple Data / tek komut çok veri) ve vektörizasyon teknolojilerini kullanabilmek için işlemlerin mümkün olduğu kadar matris veya vektörler üzerinde çalışacak şekilde düzenlenmesi (Şekil 3.9), performans açısından önemli olmuştur.



Şekil 3.9. Chetlur ve arkadaşları tarafından önerilen konvolüsyon indirgeme metodu (Chetlur ve ark., 2014)

Buna göre bir katmandaki her bir filtre ve görüntüye ait her bir kanal, Şekil 3.9’da görüldüğü gibi, ikinci matris ters çevrilip çarpıldığında, filtreler görüntü üzerinde düşmeleri gereken konumlara düşecek şekilde iki matrise yerleştirilir. Böylece yalnızca bu iki matrisin çarpımı, tüm filtreler, tüm kanallarda gezdirildiğinde elde edilecek çarpım olacaktır.

Bu yöntemin bellek kullanımını katlaması gibi bir dezavantajı bulunmaktaysa da sıralı pek çok işlemi tek bir matris çarpımına indirgediği için performansı arttırması beklenmektedir.

Üçüncü implementasyonda (Conv_3), çarpımlar tek bir matris çarpımına indirgenirken, döngü içlerinde tekrarlı olarak yapılan hedef matristeki indis hesaplamaları kaldırılmıştır.

Dördüncü implementasyonda (Conv), üçüncü implementasyonun döngüleri paralelleştirilmiş ve son durumda CNNFET’te konvolüsyon için bu implementasyon kullanılmıştır.

Bu dört implementasyonun, bir görüntünün ağdan geçirilme hızına katkısı bir sonraki ana bölümde incelenecektir.

3.5.3. Paralleleştirme

CNNFET, veri setleri üzerinde çalışacağı için aynı anda birden fazla görüntüyü yükleyip işlemesi mümkündür. Ancak birden fazla görüntünün aynı anda bellekteki aynı ağ yapısından geçirilmesi mümkün değildir. Katmanlardan birinden bir görüntü geçerken aynı anda başka bir işlemci çekirdeğinde çalışan bir iş parçacığı, bir diğer iş parçacığının yazdığı bellek alanına kendi sonuçlarını yazabilir. Bu durumda ağdan geçen hiçbir görüntü için ağ çıktısı doğru olmayacaktır.

İmplementasyon aşamasında her katmanın, kendinden sonraki katmanın giriş tensörünü kendi çıkış tensörü olarak kullanması, bellek gereksinimi açısından büyük bir avantaj sağlasa da birden fazla görüntünün ardışık iki katmandan geçmesini de imkânsız hale getirmektedir.

Paralleleştirmeyi mümkün kılmak için, ağ yapısının paralel yürüyecek iş parçacığı sayısı kadar klonlanması ve her iş parçacığının ayrı bir klon üzerinde çalışması mümkündür. Ancak konvolüsyonel yapay sinir ağlarının bellekte kapladığı alan fazlaca büyük olduğu için bu çözüm, bellek kullanımı nedeniyle pratik değildir.

Çalışma kapsamında bu probleme, bellek gereksinimini arttırmadan, her katmanda bir tek mutex kullanarak bir çözüm sunulmuştur. Buna göre her iş parçacığı n . katman üzerinde çalışırken, bulunduğu kamanın ve bir sonraki katmanın $(n+1)$ tensörüne erişimi kilitler. Böylece bir sonraki katmana geçene kadar, kendi üstünde çalıştığı katmanın ayırdığı bellek

alanında yalnızca kendisi işlem yapar. İş parçacığı sonraki katmana geçtiğinde, zaten geçtiği katmanın tensörüne erişimi bir önceki katmandayken kilitlemiştir. n . katmanın kilidini açarak bekleyen iş parçacıklarından birinin girişine izin verir. Sonrasında $(n+2)$. katmanın tensörünü kilitler ve bulunduğu katmanda $(n+1)$ yer alan giriş tensörünü işlemeye başlar. Böylece her iş parçacığı, bir katman üzerinde yalnızca 2 katmanı kilitleyerek işlemlerini sürdürür.

Bu sayede ağı bellekte klonlanmasına ve her iş parçacığının, başka bir iş parçacığını işlemekle görevli olduğu görüntüyü tüm ağdan geçirene kadar beklemesi gerekmeyecektir. CNNFET, bu yöntemle fiziksel çekirdek sayısı kadar iş parçacığı çalıştırıp görüntüleri paralel olarak yükleyerek bellek aynı ağ yapısından sorunsuz ve oldukça hızlı bir şekilde geçirmektedir. Parallelleştirme, CNNFET içerisinde sınıflandırma aşamasında uygulanmaktadır. Ancak, öznitelik vektörleri birbirlerinden bağımsız olduğu için yalnızca sınıflandırma başarısını belirleyecek olan ve ortak kullanılan karmaşıklık matrisinin üzerinde işlem yapılan bölge kontrol altında tutulmuştur.

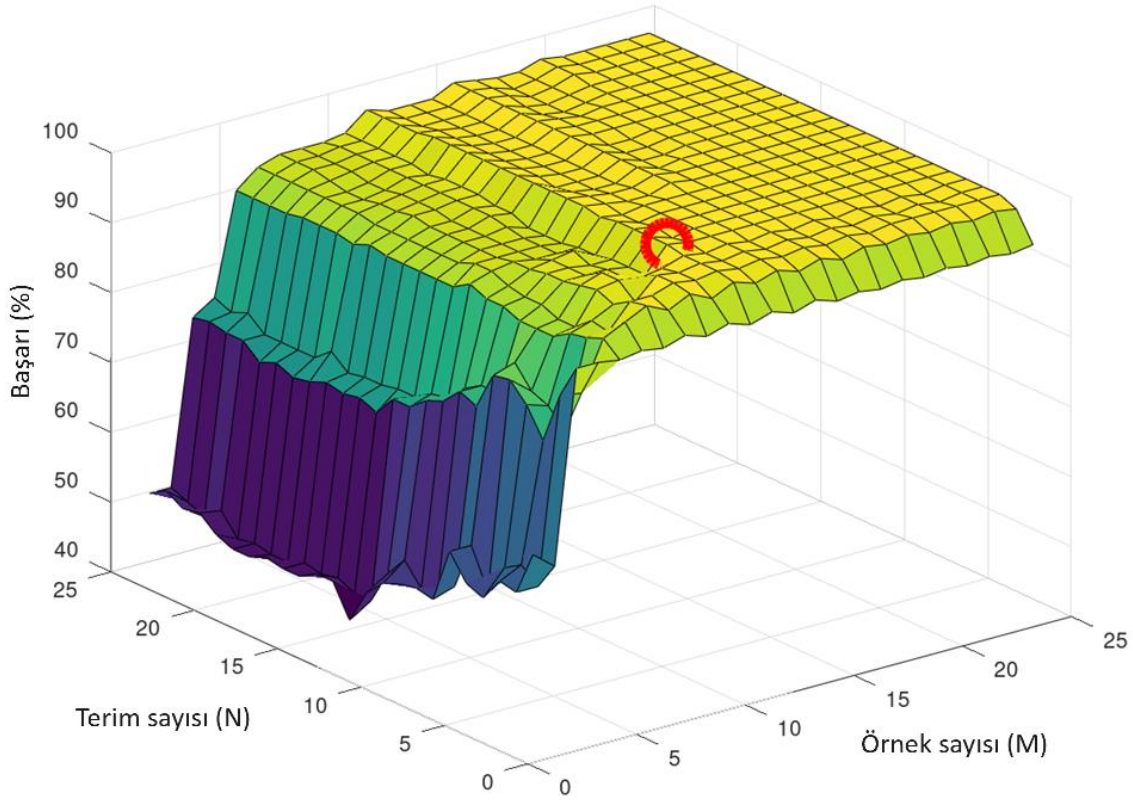
Parallelleştirme, CNNFET içerisinde sınıflandırma aşamasında uygulanmaktadır. Bir görüntü bir konvolüsyonel yapay sinir ağından geçerken en fazla zaman harcanan yer konvolüsyon katmanıdır. Bunun sebebi, çok sayıda konvolüsyon filtresinin, ağ derinleştikçe kanal sayısı artan bir görüntü üzerinde gezdirilmesinin gerekliliğidir.

4. DENEYSEL ÇALIŞMALAR

Çalışma kapsamında Fourier serileri yönteminde parametrelerin sonuca etkisinin incelenmesi amacıyla her iki parametrenin farklı kombinasyonları ile testler yapılarak bir çalışma yürütülmüştür. Sonrasında yazılan uygulama, çeşitli veri setleri üzerinde çalıştırılarak uygulamanın etkinliği incelenmiştir.

4.1. Fourier Serileri ve Parametrelerin Etkileri

Flavia veri setinden elde edilen alt set üzerinde toplamda 576 N ve M kombinasyonu test edilmiştir. Her bir N , M ikilisi için ayrı ayrı 10 parçalı çapraz doğrulama yapılmıştır. Elde edilen sınıflandırma başarıları Şekil 4.1 ve Çizelge 4.1’de verilmiştir.



Şekil 4.1. Test edilen N ve M değerleri için elde edilen sınıflandırma başarıları ve en yüksek sınıflandırma başarısını (%98,5) veren en küçük N ve M değerleri

En uygun parametreler, en iyi sonucu veren mümkün olan en düşük değerlerdir. En yüksek başarımlar $N=7$ ve $M=12$ olarak alındığında 98,5% olarak ölçülmüştür. Bu durumda en az 7 sinüs ve kosinüs terimi yaparak sınıflarını ayırt etmek için yeterli olmuştur. Ayrıca yeniden

yapılandırma aşamasında her sinyalden 12 örnek toplanmasının, yaprakların ayırt edici bir biçimde temsili için uygun olduğu görülmektedir.

Belli bir noktadan sonra N ve M değerlerini arttırmanın sınıflandırma sonucuna pozitif bir katkısı bulunmamaktadır. Bu nedenle bu değerlerin, en uygun değerlerin çok fazla üzerinde seçilmesi yalnızca işlem yükünü arttıracaktır.

Çizelge 4.1. N (terim sayısı) ve M (örnek sayısı) parametreleri 2 ile 25 arasında değerler aldığıında elde edilen sınıflandırma başarıları (%)

$M \backslash N$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	68	65,5	93	93	92,5	93	92,5	93,5	92,5	93,5	92,5	93,5	92,5	93,5	92,5	93	92,5	93	92,5	93	92,5	93	92,5	93
3	61,5	61,5	88	87,5	94,5	96	97,5	96	97,5	96	97,5	96,5	97,5	96,5	97,5	96,5	97,5	97	97,5	97	97,5	97	97,5	97
4	62	63,5	80,5	81	93,5	93,5	97	97	98	97,5	98	97,5	98	97,5	98	97,5	98	97,5	98	98	98	98	98	98
5	58,5	62	84,5	83,5	93	94	95,5	95	98	97,5	98	97,5	98	97,5	98	97,5	98	97,5	98	97,5	98	97,5	98	98
6	59,5	57,5	87	84,5	91,5	92	96,5	97	96,5	97	98	97,5	98	97,5	98	97,5	98	97,5	98	97,5	98	97,5	98	98
7	63	63	86,5	87,5	91,5	92	96	97	95,5	95	98,5	98	98	98	98	98	98	98	98	98	98	98	98	98
8	55	56	81	82	91,5	93	95,5	95,5	95,5	95	98,5	97,5	98	98	98	98	98	98	98	98	98	98	98	98
9	55,5	54,5	82	82	92,5	93	95,5	95,5	95	95	98,5	98	98	98	98	98	98	98	98	98	98	98	98	98
10	55	53,5	79,5	80,5	92,5	93,5	95	95,5	96	94,5	98,5	97,5	98	98	98	98	98	98	98	98	98	98	98	98
11	54	56,5	78,5	78	93,5	94,5	95	94,5	94,5	94,5	98	97,5	98	98	98	98	98	98	98	98	98	98	98	98
12	51,5	53,5	76,5	77,5	92	94,5	94,5	94,5	94,5	94	98,5	98	97,5	97,5	98,5	98	98	98	98	98	98	98	98	98
13	46,5	48,5	76,5	76	92	93,5	94,5	94,5	94,5	94	98,5	98	97,5	97,5	98,5	98	98	98	98	98	98	98	98	98
14	51	48,5	73,5	76	92	93,5	94,5	94,5	95	94	98,5	97,5	97	97	98,5	97,5	98	98	98	98	98	98	98	98
15	51	49,5	74,5	75,5	92	93,5	94,5	94,5	95,5	95	98,5	97	97	97	98,5	97,5	97,5	98	98	98	98	98	98	98
16	49,5	49	74	75,5	92	93,5	94,5	94,5	95,5	95	98,5	97,5	97	97	98,5	97,5	97,5	98	98	98	98	98	98	98
17	49,5	49,5	74,5	75	92,5	93,5	95	95	95,5	95,5	98,5	97,5	97	97	98,5	97,5	97,5	98	98	98	98	98	98	98
18	50	48,5	73,5	77	91,5	93,5	94,5	95	95,5	95,5	98,5	97,5	97	97	98,5	98	98	98	98	98	98	98	98	98
19	49	48,5	73	75	92	94	94,5	95	95,5	95,5	98,5	97,5	97	97	98,5	98	98	98	98	98	98	98	98	98
20	49	49	74	75	92	94	94	94,5	95,5	95	98,5	97,5	97	97	98,5	98	97,5	98	98	98	98	98	98	98
21	50	48,5	73	75	92	94	94	94,5	95,5	95,5	98,5	97,5	97	97	98,5	97,5	97	97,5	98	98	98	98	98	98
22	52	51,5	75	75	91,5	94	94,5	95	95,5	95,5	98	97,5	97	97	98,5	97,5	97	97,5	98	98	98	98	98	98
23	51,5	49,5	75	77	91,5	94	95	95	95,5	95,5	98	97	97	97	98,5	98	97	97,5	98,5	98	98	98	98	98
24	53	50,5	75,5	76	91,5	94	95	95	95,5	95,5	98	97	97	97	98,5	98	97	97,5	98,5	98	98	98	98	98
25	51,5	51,5	75	77	91,5	94	95	95	95	95,5	98	97	97	97	98,5	98	97,5	97,5	98,5	98	98	98	98	98

Sonuçlar, örnek sayısı en uygun değer altında seçildiğinde, terim sayısını arttırmanın başarıyı düşürdüğünü de göstermektedir. Daha az örnek toplamak, örnekleme frekansını düşürmek anlamına geleceğinden, örnekler asıl konturu temsil edememektedir. Diğer yandan, terim sayısını arttırmak, Fourier serilerinin daha çok detay yakalamasını sağlamaktadır. Ancak her bir sinüs-kosinüs terimi, bir öncekinden daha yüksek bir frekans

bilgisi taşıyabildiğinden çok fazla terim kullanmak, sonuçların gürültüye daha hassas hale gelmesine sebep olacaktır.

Her bir yaprak kartezyen koordinat sisteminde 1000'den fazla nokta ile temsil ediliyorken, Fourier serileri yardımıyla yalnızca yaprak başına 12 adet sayının yaprakları başarıyla temsil edebildiği görülmüştür. Ayrıca yapraklar üzerinde kullanılabilecek optimum değerlerin de tespiti sağlanmıştır. Fourier serileri yalnızca yaprakları temsil eden değerlerin sayısının çok fazla düşürülmesini değil aynı zamanda sabit olmasını da sağlamaktadır. Konturlar her yaprak için farklı sayıda nokta taşıyorken, Fourier serileri yardımıyla yeniden yapılandırma sonunda her yaprak diğerleri ile eşit sayıda değer ile temsil edilmektedir.

CNNFET ile elde edilen sonuçların karşılaştırılabilmesi için, en iyi parametre ikilisiyle bütün veri seti üzerinde test tekrarlandığında, tür sayısı artması nedeniyle sınıflandırma başarısı %75,197'ye düşmüştür.

4.2. CNNFET ile Ağın Tamamının Öznitelik Çıkarıcı Olarak Kullanıldığı Çalışmalar

CNNFET'in işlevleri, daha önce tanıtımı yapılan veri setleri üzerinde test edilmiştir. Testlerde, geliştirilen CNNFET araç seti yardımıyla örnekler ön eğitilmiş yapay sinir ağından geçirilip son katmandan çıkış alınarak, alınan çıkışlar öznitelik vektörleri olarak değerlendirilmiştir. Testlerin tamamı, 10 parçalı çapraz doğrulama yöntemi ile yürütülmüş ve sonuçlar doğru sınıflandırılan örneklerin, toplam örnek sayısına oranı olarak sunulmuştur.

4.2.1. Flavia yaprak görüntüleri veri seti

Flavia veri setinde yer alan 32 farklı türe ait 1907 yaprak görüntüsü, hiçbir ön işlem uygulanmadan, CNNFET yardımıyla öznitelik ayıklama işlemine tabi tutulmuş ve elde edilen öznitelikler herhangi bir ön işlem uygulanmaksızın, $k=3$ alınarak en yakın k -en komşu algoritması ile sınıflandırılmıştır. Elde edilen sınıflandırma başarısı %96,749 olmuştur. Ancak veri setinde yer alan görüntülerin kategorileri farklı bir dosyada verildiğinden, kategoriler, öznitelik ayıklama işleminden sonra düzenlenmiştir.

Aynı özniteliklerle veri seti, doğrusal çekirdek kullanılarak destek vektör makineleri ile sınıflandırıldığında başarı %98,899 olarak ölçülmüştür. Elde edilen karmaşıklık matrisi Çizelge 4.2'de verilmiştir.

Çizelge 4.2. Flavia veri setinde elde edilen karmaşıklık matrisi

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	aa	ab	ac	ad	ae	af	<-- sınıflandırma sonucu		
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	a = anhui barberry	
0	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b = beales barberry	
0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	c = big-fruited holly	
0	0	0	65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d = camphortree	
0	0	0	0	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	e = canadian poplar	
0	0	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f = castor aralia	
1	0	0	0	0	0	54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	g = chinese cinnamon	
0	0	0	0	0	0	0	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	h = chinese horse chestnut
0	0	0	0	0	0	0	0	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	i = chinese redbud
0	0	0	0	0	0	0	0	0	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	j = chinese toon
0	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	k = chinese tulip tree
0	0	0	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	l = crape myrtle, crepe myrtle
0	0	0	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	m = deodar
0	0	0	0	0	0	1	0	0	0	0	0	0	51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n = ford woodlotus
0	0	0	0	0	0	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	o = ginkgo, maidenhair tree
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p = flossy privet
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	q = goldenrain tree
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r = japan arrowwood
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	s = japanese cheesewood
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0	0	0	0	0	1	0	0	t = japanese flowering cherry
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	u = japanese maple
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	v = nanmu
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	66	0	0	0	0	0	0	0	0	0	0	0	0	w = oleander
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0	0	0	0	0	x = peach
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	3	0	y = pubescent bamboo
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0	z = southern magnolia
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	aa = sweet osmanthus
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	ab = tangerine
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	ac = trident maple
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73	0	0	0	0	ad = true indigo
0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	ae = wintersweet
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	af = yew plum pine

Lui ve arkadaşları, (Lui ve ark., 2018) çalışmalarında, verileri arttırarak kendi eğittikleri bir derin yapay sinir ağı ile Flavia veri seti üzerinde %87,92'lik bir sınıflandırma başarısı elde etmişlerdir.

4.2.2. IQ-OTH_NCCD tomografi görüntüleri veri seti

IQ-OTH_NCCD akciğer tomografileri veri seti üzerinde yapılan çalışmada CNMFET kullanılarak yapılan çalışmada 1097 adet örneğin 1053 tanesi doğru sınıflandırılarak, %95,989'luk bir başarı elde edilmiştir. Sınıflandırma sonrası aşağıdaki karmaşıklık matrisi elde edilmiştir (Çizelge 4.3).

Çizelge 4.3. IQ-OTH_NCCD veri setinde elde edilen karmaşıklık matrisi

a	b	c	<-- sınıflandırma sonucu
98	3	19	a = İyi huylu
3	556	2	b = Kötü huylu
11	6	399	c = Normal

Aynı veri seti üzerinde Al-Yasriy ve arkadaşlarının, veri setinin %70'ini eğitime, %30'unu teste ayırarak elde ettiği sınıflandırma başarısı %93,548 olarak raporlanmıştır (Al-Yasriy ve ark., 2020). Bu değeri, çalışmalarında, önceden eğitilmemiş bir derin yapay sinir ağı modelini eğitip sonrasında sınıflandırma için kullanarak elde etmişlerdir.

4.2.3. 102flowers çiçek görüntüleri veri seti

Bir diğer veri seti olan 102flowers veri setinde en az 100 görüntü içeren 23 kategorideki 3375 adet görüntüden 3071 tanesi doğru sınıflandırılmıştır. Böylece sınıflandırma başarısı %90,993 olmuştur. Sınıflandırma sonunda aşağıdaki karmaşıklık matrisi elde edilmiştir (Çizelge 4.4).

Çizelge 4.4. 102flowers veri setinde elde edilen karmaşıklık matrisi

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	<-- sınıflandırma s.
107	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	a = class37
0	125	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	b = class41
0	0	102	2	6	0	0	0	0	0	5	1	0	1	0	0	0	3	2	2	1	2	3	c = class43
0	0	0	192	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	d = class46
0	0	6	1	228	1	0	0	1	0	0	1	2	1	0	0	0	4	7	1	1	2	2	e = class51
0	0	0	2	0	107	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f = class56
0	0	0	0	0	0	111	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	g = class58
0	0	0	0	0	0	0	109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	h = class60
0	0	0	0	2	0	0	0	99	0	0	1	0	0	0	0	0	0	0	0	0	0	0	i = class65
0	0	0	0	0	0	0	0	0	184	2	1	0	0	3	0	2	2	0	0	0	0	0	j = class73
0	0	1	2	3	0	2	0	1	0	146	0	0	0	2	5	2	0	0	2	3	0	2	k = class74
0	0	0	0	2	0	0	0	0	0	0	117	1	0	0	0	0	0	0	0	0	0	0	l = class75
0	0	2	0	9	0	0	0	0	0	0	3	89	0	1	1	0	0	1	1	0	0	0	m = class76
0	1	0	0	2	0	0	0	0	0	0	0	0	244	1	0	0	3	0	0	0	0	0	n = class77
0	0	0	0	0	0	0	0	0	1	2	0	1	0	125	1	1	1	0	3	0	0	2	o = class78
0	0	1	0	0	0	0	0	0	1	6	1	0	0	3	87	0	1	0	2	1	1	1	p = class80
0	0	0	1	0	0	0	0	0	0	2	0	0	0	5	0	157	0	1	0	0	0	0	q = class81
0	0	1	0	3	0	0	0	0	3	0	1	0	3	1	0	0	91	1	1	0	4	3	r = class82
0	1	1	0	10	0	1	0	0	0	2	0	0	0	0	0	0	0	107	0	8	0	1	s = class83
1	0	2	0	2	0	0	0	0	0	2	1	0	0	1	2	0	1	0	138	0	0	4	t = class88
0	0	1	3	2	0	0	1	1	0	2	0	0	0	1	0	0	0	5	0	168	0	0	u = class89
0	0	3	0	5	0	0	0	0	0	1	1	0	0	0	1	1	2	0	0	0	145	3	v = class94
0	0	4	1	9	0	0	0	0	1	7	0	1	0	4	0	1	2	0	3	1	1	93	w = class95

Nilsberg ve Zisserman, veri setlerini sundukları ve farklı öznitelikler kullanarak, tüm kategoriler üzerinde yürüttükleri çalışmalarında, %43,0 – %72,8 arasında değişen sınıflandırma başarıları yakalamıştır (Nilsberg ve Zisserman, 2008).

Aynı test tüm kategoriler üzerinde, ancak herhangi bir öznitelik ayıklama veya ön işlem yapılmadan tekrarlandığında ise CNNFET üzerinde, 102 kategoride, 8189 örnekten 5542 tanesi doğru sınıflandırılarak başarı %67,676'ye düşmektedir.

4.2.4. QUT balık veri seti

QUT balık veri setinde CNNFET ile öznitelik çıkarımı yapıldıktan sonra destek vektör makineleri ile yapılan sınıflandırmanın sonucunda 401 adet görüntünün 318 tanesi doğru sınıflandırılmış ve başarı % 79.302 olarak ölçülmüştür. Elde edilen karmaşıklık matrisi Çizelge 4.5'te sunulmuştur.

Çizelge 4.5. QUT veri setinde elde edilen karmaşıklık matrisi

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	<-- sınıflandırma sonucu
17	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	a = cantherhines_dumerilii
0	20	1	0	0	0	0	0	2	0	0	0	0	0	0	0	1	0	b = cantherhines_pardalis
0	2	22	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	c = caranx_melampygus
0	0	0	18	0	0	0	1	1	0	0	0	0	0	0	0	1	0	d = cheilinus_chlorourus
0	0	0	0	15	0	0	1	0	0	0	0	0	0	4	1	0	0	e = cirrhlabrus_punctatus
0	0	0	0	1	18	0	0	0	0	0	0	0	0	0	2	1	1	f = halichoeres_melanurus
0	0	0	0	0	1	19	0	1	0	0	0	0	0	0	0	0	0	g = hemigymnus_fasciatus
0	0	0	1	0	0	0	14	0	0	0	0	0	0	0	4	0	1	h = leptojulius_cyanopleura
0	1	1	0	0	0	0	1	18	1	1	0	0	0	0	0	0	0	i = lethrinus_nebulosus
0	0	0	0	0	0	0	0	1	15	2	2	0	0	0	1	1	0	j = lutjanus_bohar
0	0	0	0	0	0	0	1	0	2	16	0	0	0	0	2	1	0	k = lutjanus_fulviflamma
0	0	0	0	0	0	0	0	0	1	0	19	1	0	0	0	0	0	l = lutjanus_sebae
0	1	0	0	0	0	1	0	1	0	1	0	16	0	0	0	2	0	m = monotaxis_grandoculis
0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	0	n = oxymonacanthus_longirostris
0	0	0	0	1	0	0	0	1	1	0	0	0	0	20	0	0	0	o = pseudanthias_squamipinnis
0	0	0	1	1	0	0	4	0	1	0	0	0	0	0	15	0	0	p = stethojulis_interrupta
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	16	3	q = thalassoma_lunare
0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	1	4	14	r = thalassoma_lutescens

Literatürde veri seti üzerinde yapılan çalışmalarda belirli alt setler üzerinden çalışıldığından, aynı alt setlerin elde edilmesi ve testlerin tekrarlanması mümkün olmamıştır. Ancak Mathur ve Goel 37 türden 600 görüntü için kendi eğittikleri konvolüsyonel yapay sinir ağında, %84,92'lik bir başarı raporlamıştır (Mathur ve Goel, 2021).

4.2.5. AppleScabFDs elma görüntüleri veri seti

Veri seti üzerinde CNNFET ile son öznetelik çıkarımı yapıldıktan sonra destek vektör makineleri ile yapılan sınıflandırmanın sonucunda 297 adet görüntünün 251 tanesi doğru sınıflandırılmış ve başarı %84,512 olarak ölçülmüştür. Elde edilen karmaşıklık matrisi Çizelge 4.6’da sunulmuştur.

Çizelge 4.6. AppleScab veri setinde elde edilen karmaşıklık matrisi

a	b	<-- sınıflandırma sonucu
64	26	a = Sağlıklı
20	187	b = Hastalıklı

Nouaze ve Sikati, veri üzerinde derin öğrenme modelleri kullanarak çeşitli çalışmalar yapmışlardır. Elde ettikleri en yüksek başarıyı kendi eğittikleri derin öğrenme modeli ile elde etmişlerdir (Nouaze ve Sikati, 2023). En başarılı sonuçlar incelendiğinde, kesinlik, duyarlılık ve F1 değerleri sırasıyla %94,8, %85,20 ve %89.75 olarak elde edilmiştir. Karşılaştırma yapılabilmesi amacıyla, CNNFET ile elde edilen öznetelikler kullanılarak yapılan çalışmada, karmaşıklık matrisi üzerinde aynı metrikler, %84,3, %84,5 ve %84,4 olarak hesaplanmıştır. Bu değerler, Nouaze ve Sikati tarafından elde edilen değerlerin gerisindedir. Tez çalışmasının ilerleyen kısımlarında, katmanlar üzerinde yapılan daha detaylı çalışma sonuçları ile birlikte bu sonuçlar yeniden değerlendirilecektir.

4.3. Konvolüsyon İmplementasyonlarının ve Paralleleştirmenin Etkileri

Konvolüsyon işlemleri için yazılan dört farklı implementasyonun ağdan bir görüntünün geçme hızına etkileri, 8 fiziksel, 4 sanal çekirdeği olan Intel Core i5-6500U 2.50GHz işlemci ve 8GB RAM üzerinde karşılaştırılmıştır.

Conv_2 implementasyonu ilkinde göre çok fazla hızlı çalışmamıştır. Ancak konvolüsyon filtrelerini tek bir matris çarpımına indirgeme yöntemi sonraki implementasyonlarda hızın artırılmasına yardımcı olmuştur. Conv_3, Conv_2’den yaklaşık 3,5 kat, Conv_1’den 3,8 kat daha hızlı çalışmıştır. Bir sonraki implementasyon (Conv), 2 fiziksel (4 sanal) işlemci çekirdeği üzerinde yapılan paralelleştirmeye birlikte Conv_1’den 7,3 kat, Conv_2’den 6,5 kat, Conv_3’ten 1,9 kat daha hızlı çalışmıştır.

Matris çarpımlarının BLAS desteğiyle yapılması durumunda ise, elde edilen en iyi sonuçta yaklaşık 6,8 katlık bir hız artışı sağlamıştır. Böylece konvolüsyonu matris çarpımına indirgeme ve BLAS desteği son durumda, ilk implementasyondan yaklaşık 49,9 kat daha hızlı işlem yapılmasını sağlamıştır. İşlemci çekirdek sayısının daha fazla olması, görüntülerin ağdan geçiş hızının katlanarak artmasını sağlayacaktır. Ayrıca görüntü sayısının yüksek olması da artış oranının artmasına sebep olacaktır.

4.4. Farklı Derinliklerde Elde Edilen Özniteliklerin ve Sonuçlara Etkilerinin Analizi

CNNFET kullanılarak, ön eğitilmiş konvolüsyonel ağların herhangi bir katmanından alınan sonuçlar kullanılabilir. Tez çalışması kapsamında, ön eğitilmiş konvolüsyonel yapay sinir ağlarının farklı katmanlarında, öznitelik çıkarımı yapılan görüntülere özel daha ayırt edici öznitelikler yakalanabileceği öngörüsünü doğrulamak amacıyla veri setleri üzerinde farklı katmanlardan elde edilen özniteliklerle çalışmalar yürütülmüştür. Çalışmalar sırasında yine bir VGG-f modelinde olan “imagenet-matconvnet-vgg-f.cenin” ön eğitilmiş ağı kullanılmıştır. Kullanılan bu ağ modelinde 8 konvolüsyon katmanı mevcuttur (Çizelge 4.7).

Çizelge 4.7. Konvolüsyon katmanları, derinlikleri ve çıkış sayıları

Katman	Derinlik	Çıkış Sayısı
Konvolüsyon	1	186624
Konvolüsyon	4	186624
Konvolüsyon	7	43264
Konvolüsyon	9	43264
Konvolüsyon	11	43264
Konvolüsyon	14	4096
Konvolüsyon	16	4096
Konvolüsyon	18	1000

CNNFET ile kullanılan ön eğitilmiş ağın her bir katmanından ayrı ayrı öznitelik setleri elde edilmiştir. Elde edilen özniteliklerin veri setlerinin sınıfları ile mutlak Pearson korelasyonları incelenmiştir. Daha sonra bu öznitelikler içerisinde, ağın çıkış sayısı kadar öznitelik (1000) en yüksek korelasyon değerlerine göre seçilip doğrusal çekirdek ile birlikte

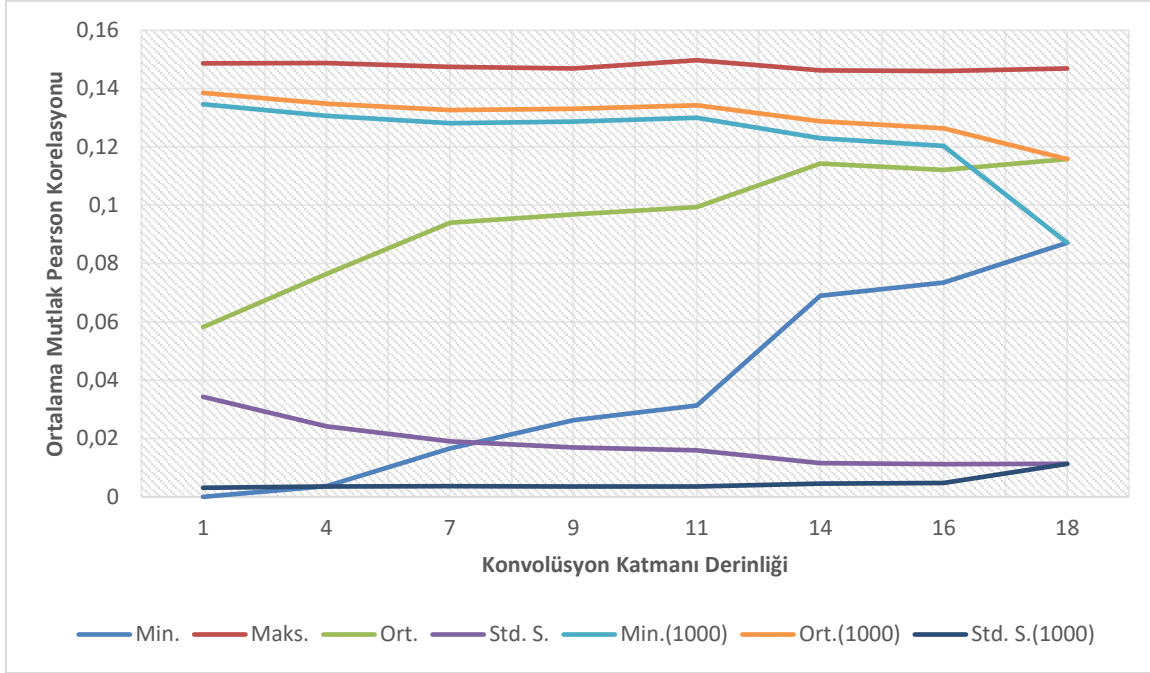
destek vektör makineleri kullanılarak ayrı ayrı sınıflandırma yapılmış ve özniteliklerin sınıflandırma başarıları ve dolayısıyla veri setlerini temsil yetenekleri değerlendirilmiştir.

4.4.1. Flavia yaprak görüntüleri veri seti

CNNFET kullanılarak Flavia yaprak görüntüleri veri setindeki her bir görüntü için, ayrı ayrı her bir katmandan belirtilen çıkış sayısı kadar öznitelik çıkarımı yapıldıktan sonra, öznitelikler ve sınıf vektörleri arasında hesaplanan ortalama mutlak Pearson korelasyonları üzerinde hesaplanan istatistiksel veriler, tümü ve en iyi 1000 değer için ayrı ayrı Çizelge 4.8’de ve Şekil 4.2’de verilmiştir. En iyi 1000 değer, korelasyonlar büyükten küçüğe doğru sıralanarak elde edilen listeden alınan ilk 1000 değerdir.

Çizelge 4.8. Flavia veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Derinlik	Ortalama Mutlak Pearson Korelasyonları						
	Tümü				En İyi 1000 Öznitelik		
	Min.	Maks.	Ort.	Std. S.	Min.	Ort.	Std. S.
1	0,0000	0,1486	0,0582	0,0343	0,1346	0,1385	0,0031
4	0,0036	0,1487	0,0764	0,0242	0,1306	0,1348	0,0035
7	0,0166	0,1474	0,0940	0,0190	0,1281	0,1326	0,0036
9	0,0263	0,1469	0,0968	0,0169	0,1287	0,1330	0,0035
11	0,0313	0,1497	0,0993	0,0159	0,1300	0,1343	0,0035
14	0,0689	0,1462	0,1143	0,0116	0,1229	0,1288	0,0045
16	0,0734	0,1460	0,1121	0,0112	0,1203	0,1264	0,0047
18	0,0871	0,1469	0,1158	0,0113	0,0871	0,1158	0,0113



Şekil 4.2. Flavia veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

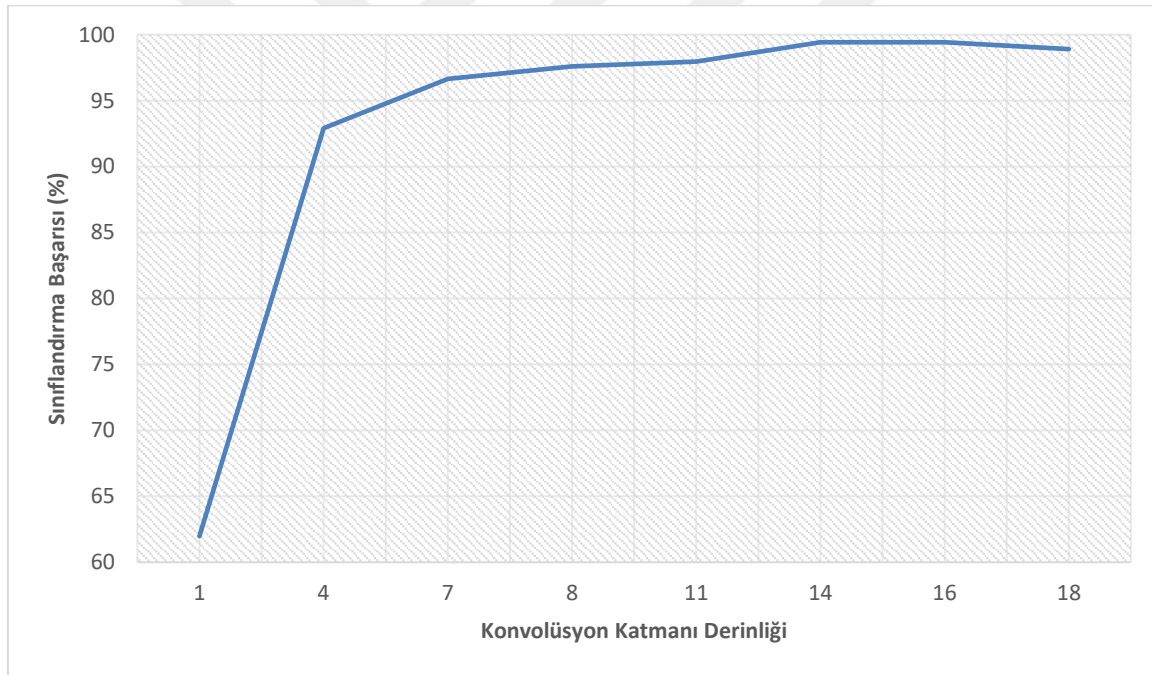
Çizelge 4.8 ve Şekil 4.2 birlikte incelendiğinde, maksimum korelasyon değerlerinin katmanlar boyunca çok fazla değişmediği, ortalamanınsa yükseldiği görülmektedir. Ancak son katmana doğru ilk 1000 değer için ortalama ve minimum değerlerde düşüş meydana gelmiştir. Tüm özniteliklerin standart sapmasında da katmanlar ilerledikçe, öznitelik sayılarının sayısındaki düşüş nedeniyle bir düşüş meydana gelmiştir.

Özniteliklerin sınıflarla korelasyonu incelendikten sonra en iyi 1000 adet öznitelik ile 10 parçalı çapraz doğrulama yapılarak hesaplanan sınıflandırma başarıları Çizelge 4.9’da ve Şekil 4.3’te verilmiştir.

Sınıflandırma başarıları incelendiğinde (Çizelge 4.9, Şekil 4.3), ilk konvolüsyon katmanı hariç, bir önceki adımda en iyi 1000 öznitelik için elde edilen minimum ve maksimum korelasyon değerleri ile örtüşen sonuçlar alınmıştır. En yüksek değer, 14 ve 16. katmandan alınan özniteliklerle elde edilmiştir. Bu veri seti üzerinde sınıflandırma başarısı, tüm ağ kullanıldığında da zaten oldukça yüksek elde edilmiştir. Ancak 14. ve 16. ara katmanlardaki öznitelikler, tüm ağın çıkışından alınan özniteliklerden biraz daha başarılı bir sonuç vermiştir. 14. katmandan alınan özniteliklerle yapılan sınıflandırma sonucundan elde edilen karmaşıklık matrisi Çizelge 4.10’da verilmiştir.

Çizelge 4.9. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle Flavia veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

Derinlik	Sınıflandırma Başarıları (%)
1	61,9822
4	92,9208
7	96,6439
9	97,5878
11	97,9549
14	99,4232
16	99,4232
18	98,8988



Şekil 4.3. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle Flavia veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

İlk katmanda elde edilen korelasyon değerleri sonraki katmanlara göre çok düşük değilken, sınıflandırma başarıları ilk katmanda oldukça düşük elde edilmiştir. Bu sonuca neyin sebep olduğunun anlaşılması için özniteliklerin her bir sınıftaki başarı oranları ayrı ayrı incelenmiştir. Bu amaçla kesinlik ve duyarlılık değerlerinin harmonik ortalaması olan F1 değerlerine her bir sınıf için ayrı ayrı bakılmıştır. F1 değerlerinin belli az sayıda sınıfta yüksek olduğu ve 0,893 değerine kadar ulaşabildiği, diğer sınıflarda düşük olduğu ve 0,031

değerine kadar düştüğü görülmüştür. Bu da ilk katmandaki özniteliklerin bu veri seti üzerinde belli sınıfları, nispeten iyi temsil ederken ortalama korelasyonun yüksek çıkmasına neden olduğunu ancak diğer sınıfların temsilinde yetersiz olduğunu göstermektedir.

Çizelge 4.10. Flavia veri setinde 14. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi

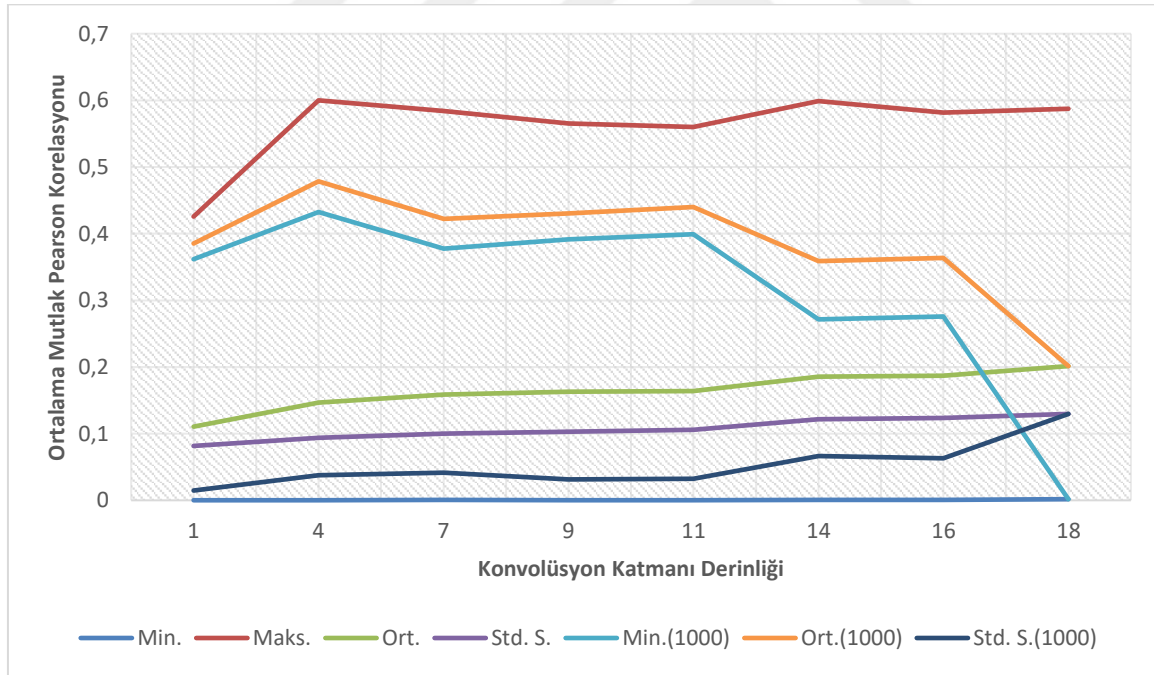
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	aa	ab	ac	ad	ae	af	<-- classified as	
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	a = anhui barberry
0	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b = beales barberry
0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	c = big-fruited holly
0	0	0	65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	d = camphortree
0	0	0	0	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	e = canadian poplar
0	0	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f = castor aralia
2	0	0	0	0	0	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	g = chinese cinnamon
0	0	0	0	0	0	1	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	h = chinese horse chestnut
0	0	0	0	0	0	0	0	0	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	i = chinese redbud
0	0	0	0	0	0	0	0	0	65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	j = chinese toon
0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	k = chinese tulip tree
0	0	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	l = crape myrtle, crepe myrtle
0	0	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	m = deodar
0	0	0	0	0	0	1	0	0	0	0	0	51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	n = ford woodlotus
0	0	0	0	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	o = ginkgo, maidenhair tree
0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	p = glossy Privet
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	q = goldenrain tree
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	r = japan arrowwood
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	s = japanese cheesewood
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	t = japanese flowering cherry
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	u = japanese maple
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	v = nanmu
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	66	0	0	0	0	0	0	0	0	0	0	0	0	w = oleander
0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	x = peach
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58	0	0	0	0	0	0	0	0	0	1	y = pubescent bamboo
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57	0	0	0	0	0	z = southern magnolia
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	aa = sweet osmanthus
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	55	0	0	0	0	ab = tangerine
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0	0	ac = trident maple
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73	0	0	0	ad = true indigo
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	ae = wintersweet
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	59	0	af = yew plum pine

4.4.2. IQ-OTH_NCCD tomografi görüntüleri veri

Aynı çalışma IQ-OTH_NCCD tomografi görüntüleri veri seti üzerinde yürütüldüğünde, elde edilen korelasyonlar üzerinde hesaplanan istatistiksel veriler, tümü ve en iyi 1000 değer için Çizelge 4.11’de ve Şekil 4.4’te verilmiştir.

Çizelge 4.11. IQ-OTH_NCCD veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Derinlik	Ortalama Mutlak Pearson Korelasyonları						
	Tümü				En İyi 1000 Öznitelik		
	Min.	Maks.	Ort.	Std. S.	Min.	Ort.	Std. S.
1	0,0001	0,4257	0,1105	0,0816	0,3618	0,3851	0,0147
4	0,0001	0,6000	0,1465	0,0937	0,4325	0,4785	0,0374
7	0,0008	0,5842	0,1587	0,0999	0,3776	0,4220	0,0414
9	0,0002	0,5652	0,1630	0,1030	0,3916	0,4304	0,0316
11	0,0003	0,5599	0,1640	0,1056	0,3992	0,4398	0,0324
14	0,0006	0,5987	0,1857	0,1217	0,2716	0,3589	0,0663
16	0,0007	0,5814	0,1868	0,1237	0,2759	0,3636	0,0630
18	0,0017	0,5876	0,2014	0,1296	0,0017	0,2014	0,1296



Şekil 4.4. IQ-OTH_NCCD veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

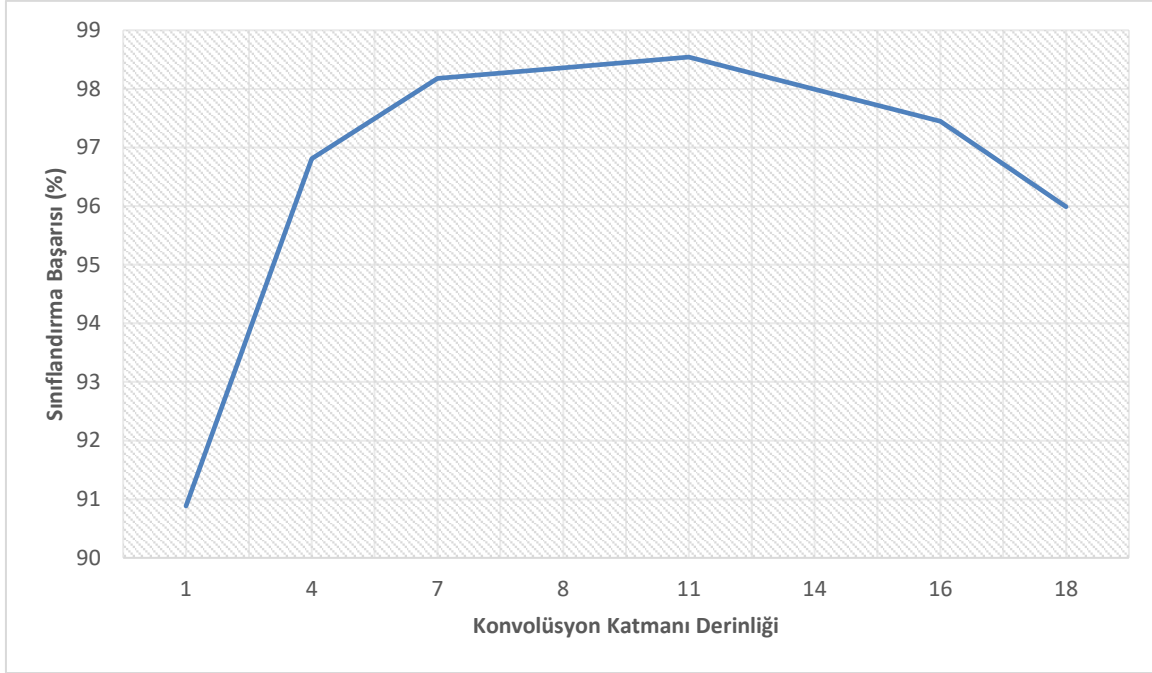
Çizelge 4.11 ve Şekil 4.4'teki veriler tüm öznitelikler için incelendiğinde, katmanlarda ilerledikçe ortalama ve standart sapmada bir artış meydana geldiği görülmektedir. En iyi 1000 öznitelik incelendiğindeyse, ortalama ve minimum korelasyon değerleri son konvolüsyon katmanında düşmektedir.

Özniteliklerin sınıflarla korelasyonları hesaplandıktan sonra en iyi 1000 öznitelik ile 10 parçalı çapraz doğrulama yapılarak elde başarı oranları Çizelge 4.12’de ve Şekil 4.5’te verilmiştir.

Çizelge 4.12. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle IQ-OTH_NCCD veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

Derinlik	Sınıflandırma Başarıları (%)
1	90,8842
4	96,8095
7	98,1768
9	98,3592
11	98,5415
14	97,9945
16	97,4476
18	95,9891

IQ-OTH_NCCD veri seti üzerindeki sınıflandırma başarıları incelendiğinde (Çizelge 4.12, Şekil 4.5), en yüksek değer, 11. katmandan alınan özniteliklerle elde edilmiştir. Katman derinliği artarken başlarda başarı yükselmiş ancak daha sonra düşmüştür. En yüksek korelasyon değeri 14. katmanda yer alıyor olsa da 14. katmandaki ortalama ve minimum korelasyonlar 11. katmandaki değerlere göre düşüş göstermiştir. Bu da 11. katmanda daha başarılı sınıflandırma sonuçları elde edilmesini sağlamıştır 11. katmandan çıkarılan özniteliklerle yapılan sınıflandırma sonunda elde edilen karmaşıklık matrisi Çizelge 4.13’te verilmiştir.



Şekil 4.5. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle IQ-OTH_NCCD veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

Çizelge 4.13. IQ-OTH_NCCD veri setinde 11. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi

a	b	c	<-- sınıflandırma sonucu
114	0	6	a = İyi huylu
0	561	0	b = Kötü huylu
7	3	406	c = Normal

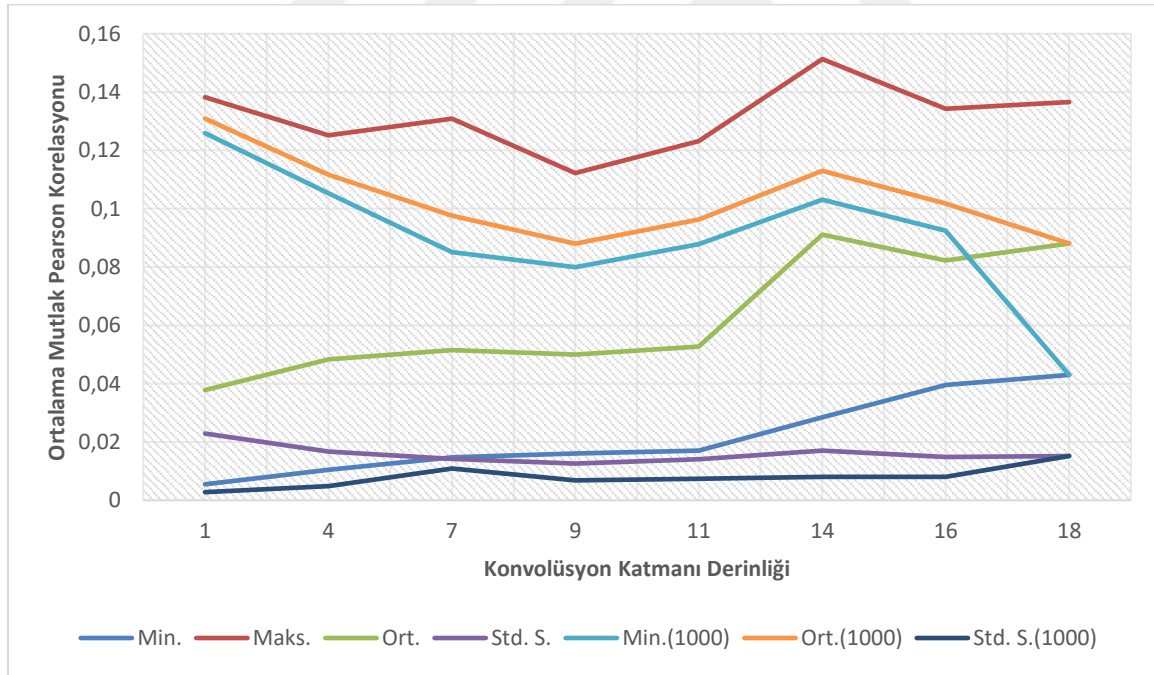
Ağın 11. katmanından çıkarılan özniteliklerle elde edilen 98,5415'lik başarı, tüm ağın öznitelik çıkarıcı olarak kullanıldığı durumda elde edilen %95,9891 değerinden belirgin olarak daha yüksektir.

4.4.3. 102flowers çiçek görüntüleri veri seti

Aynı adımlar izlenerek 102flower çiçek veri seti üzerinde ilk aşamada ortalama mutlak Pearson korelasyonları üzerinde hesaplanan istatistiksel veriler, tümü ve en iyi 1000 değer için Çizelge 4.14'te ve Şekil 4.6'de verilmiştir.

Çizelge 4.14. 102flowers veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Derinlik	Ortalama Mutlak Pearson Korelasyonları						
	Tümü				En İyi 1000 Öznitelik		
	Min.	Maks.	Ort.	Std. S.	Min.	Ort.	Std. S.
1	0,0055	0,1382	0,0378	0,0229	0,1259	0,1309	0,0028
4	0,0105	0,1251	0,0483	0,0167	0,1053	0,1116	0,0049
7	0,0147	0,1309	0,0515	0,0142	0,0851	0,0976	0,0109
9	0,0161	0,1122	0,0500	0,0126	0,0799	0,0880	0,0068
11	0,0171	0,1232	0,0527	0,0141	0,0878	0,0963	0,0074
14	0,0285	0,1513	0,0911	0,0171	0,1031	0,1130	0,0081
16	0,0395	0,1343	0,0822	0,0149	0,0925	0,1018	0,0080
18	0,0430	0,1366	0,0881	0,0152	0,0430	0,0881	0,0152



Şekil 4.6. 102flowers veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Çizelge 4.14 ve Şekil 4.6 incelendiğinde, maksimum korelasyon değerlerinin ara konvolüsyon katmanlarında özellikle 14. katmanda daha yüksek olduğu görülmektedir. Katmanlar derinleştikçe standart sapmalarda belirgin bir değişim olmamıştır. En iyi 1000 özniteliğin maksimum, ortalama ve minimum değerlerinin oluşturduğu kanala bakıldığında

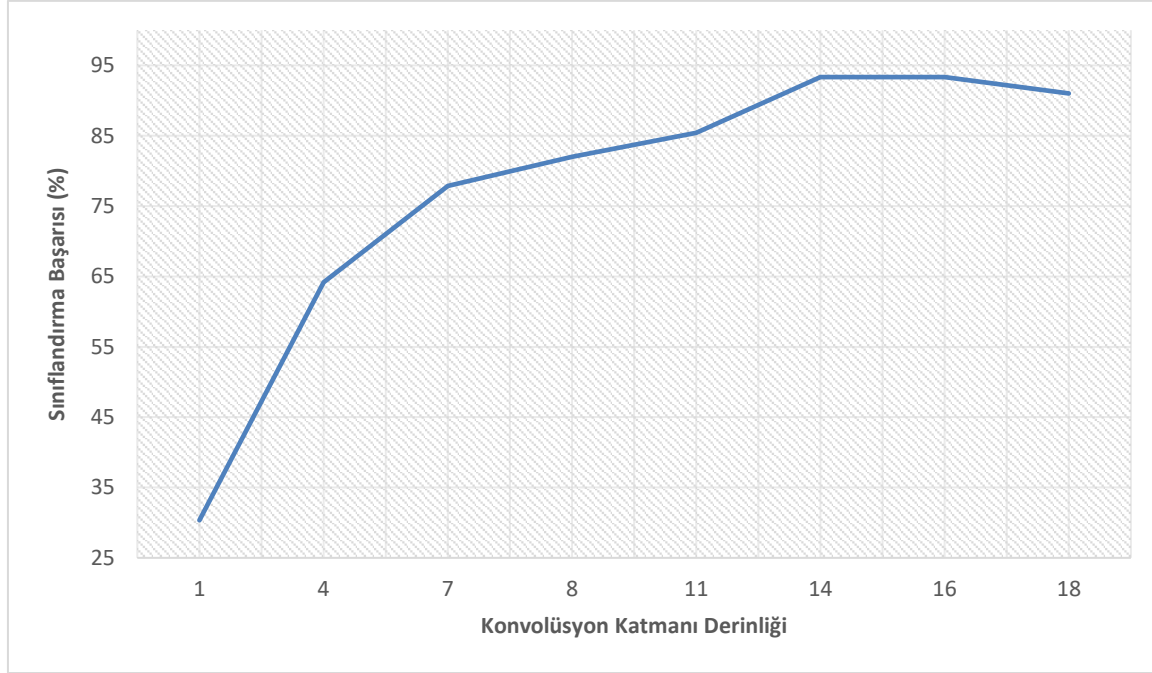
önce bir düşüş sonra bir yükseliş meydana gelmiş ve 14. katmandan sonra korelasyonlar tekrar düşmüştür.

En iyi 1000 adet öznitelik ile 10 parçalı çapraz doğrulama yapılarak elde edilen başarı oranları Çizelge 4.15'te ve Şekil 4.7'de verilmiştir.

Çizelge 4.15. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle 102flowers veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

Derinlik	Sınıflandırma Başarıları (%)
1	30,3407
4	64,1481
7	77,8667
9	81,9852
11	85,4222
14	93,3333
16	93,3333
18	90,9926

Elde edilen en iyi sınıflandırma başarısını, 14 ve 16. katmanlardan çıkarılan öznitelikler ile sağlanmıştır (Çizelge 4.15, Şekil 4.7). Başarı, 14. katmana kadar düzenli bir şekilde artış sağlamış ancak son katmanda düşmüştür. 14. katmandan elde edilen özniteliklerle yapılan sınıflandırma sonucunda elde edilen karmaşıklık matrisi Çizelge 4.16'da sunulmuştur.



Şekil 4.7. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle 102flowers veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

Çizelge 4.16. 102flowers veri setinde 14. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	<-- sınıflandırma s.
106	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	a = class37
	0	126	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	b = class41
	0	1	106	0	5	0	0	0	0	0	4	2	1	0	0	0	0	2	1	3	3	1	1	c = class43
	0	0	0	193	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	d = class46
	0	0	4	0	234	0	1	0	0	0	0	2	3	0	0	0	0	2	3	0	2	3	4	e = class51
	0	0	0	1	0	108	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	f = class56
	0	0	0	0	0	0	112	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	g = class58
	0	0	0	0	0	0	0	109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	h = class60
	0	0	0	1	0	0	0	0	99	0	2	0	0	0	0	0	0	0	0	0	0	0	0	i = class65
	0	0	1	0	0	0	0	0	0	188	1	1	0	0	2	0	0	1	0	0	0	0	0	j = class73
	0	0	1	0	1	0	1	0	2	0	153	0	0	0	1	3	2	0	0	2	2	0	3	k = class74
	0	0	0	0	3	0	0	0	0	0	0	116	1	0	0	0	0	0	0	0	0	0	0	l = class75
	0	0	0	0	5	0	0	0	0	0	0	2	98	0	2	0	0	0	0	0	0	0	0	m = class76
	0	0	0	0	0	0	0	0	0	0	0	0	0	247	0	0	0	4	0	0	0	0	0	n = class77
	0	0	0	0	0	0	0	0	0	2	2	0	1	0	126	1	2	0	0	1	0	0	2	o = class78
	0	0	1	0	1	0	0	0	0	0	4	0	0	0	1	92	0	1	0	2	1	0	2	p = class80
	0	0	0	1	0	0	0	0	0	0	1	0	0	0	2	0	161	0	0	0	0	0	1	q = class81
	0	0	1	0	7	0	0	0	0	0	0	0	1	0	1	0	0	99	1	0	0	1	1	r = class82
	0	1	3	0	6	0	1	0	0	0	0	0	1	1	1	0	0	0	112	0	5	0	0	s = class83
	0	0	4	0	0	0	0	0	0	0	2	1	0	0	0	1	0	1	0	140	0	1	4	t = class88
	0	1	3	3	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	174	0	0	u = class89
	0	0	1	0	2	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	155	1	v = class94
	0	0	5	1	9	0	0	0	0	0	5	1	1	0	2	1	0	1	0	4	1	1	96	w = class95

Ağın 14. katmanından çıkarılan özniteliklerle elde edilen %93,3333lük başarı, bu veri setinde de tüm ağın öznitelik çıkarıcı olarak kullanıldığı durumda elde edilen %90,9926'lık başarıdan daha yüksek hesaplanmıştır.

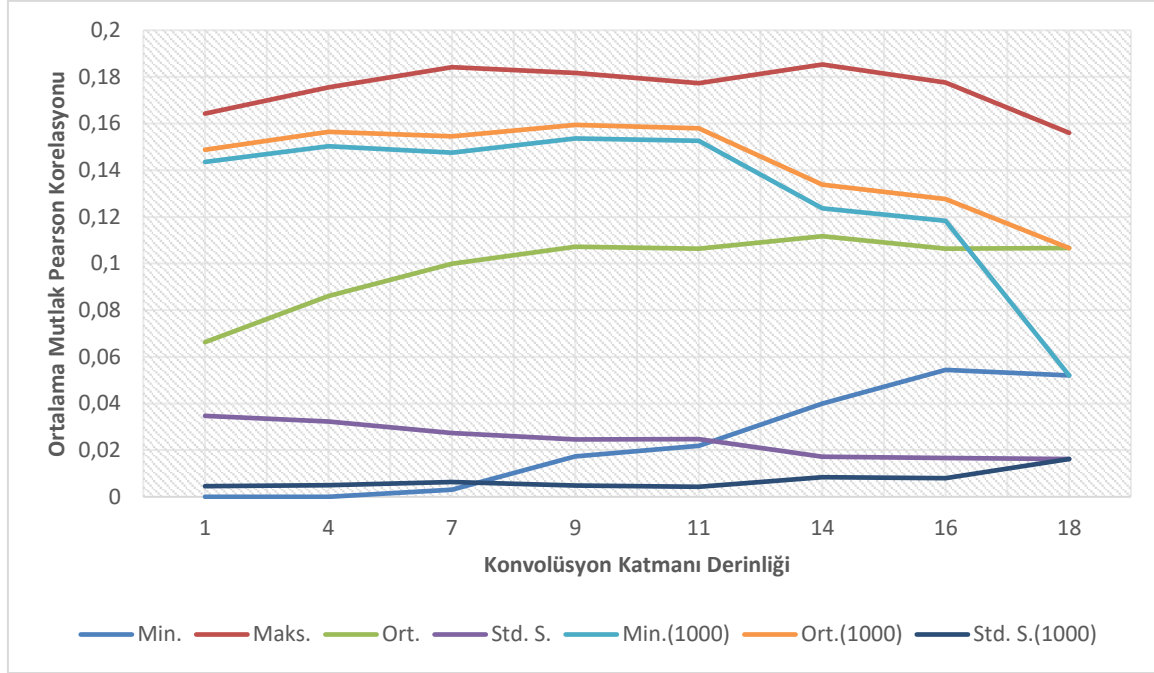
4.4.4. QUT balık görüntüleri veri seti

Aynı yöntemlerle elde edilen korelasyonlar üzerinde hesaplanan istatistiksel veriler, tümü ve en iyi 1000 değer için ayrı ayrı Çizelge 4.17'de ve Şekil 4.8'de verilmiştir.

Çizelge 4.17. QUT veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Derinlik	Ortalama Mutlak Pearson Korelasyonları						
	Tümü				En İyi 1000 Öznitelik		
	Min.	Maks.	Ort.	Std. S.	Min.	Ort.	Std. S.
1	0,0000	0,1642	0,0663	0,0347	0,1436	0,1488	0,0045
4	0,0000	0,1755	0,0860	0,0323	0,1502	0,1564	0,0050
7	0,0030	0,1842	0,0999	0,0273	0,1475	0,1545	0,0063
9	0,0173	0,1817	0,1072	0,0246	0,1536	0,1594	0,0049
11	0,0219	0,1773	0,1063	0,0247	0,1526	0,1579	0,0043
14	0,0399	0,1853	0,1117	0,0172	0,1236	0,1338	0,0084
16	0,0544	0,1775	0,1063	0,0166	0,1183	0,1276	0,0080
18	0,0520	0,1560	0,1066	0,0162	0,0520	0,1066	0,0162

Çizelge 4.17 ve Şekil 4.8 birlikte incelendiğinde, maksimum korelasyon değerlerinin ara konvolüsyon katmanlarında daha yüksek olduğu görülmektedir. Katmanlar derinleştikçe ilk 1000 değerde standart sapmalar düşerken, tüm özniteliklerde hesaplanan standart sapmalar artmaktadır. Bu da tüm değerler arasında sınıflarla daha az ilişkili daha fazla değer bulunuyor olmasından kaynaklanmaktadır. Son katmanlarda tüm özniteliklerin sınıflarla korelasyonlarının ortalaması çok fazla değişmiyorken, en iyi 1000 özniteliğin sınıflarla korelasyonu düşmektedir. Bu sonuç konvolüsyon katmanı çıkışlarının son katmanlara doğru, ön eğitilmiş ağın eğitildiği verilerle daha ilişkili olabileceken, direkt olarak ağ çıkışlarının başka sınıflandırma işlemleri için en uygun çıkışlar olmayabileceği beklentisini doğrulamaktadır. Tüm öznitelikler arasında minimum değerlere bakıldığında, derinlere inildikçe, minimum değerlerin yükseldiği ve az ilişkili özniteliklerin azaldığı görülmektedir.

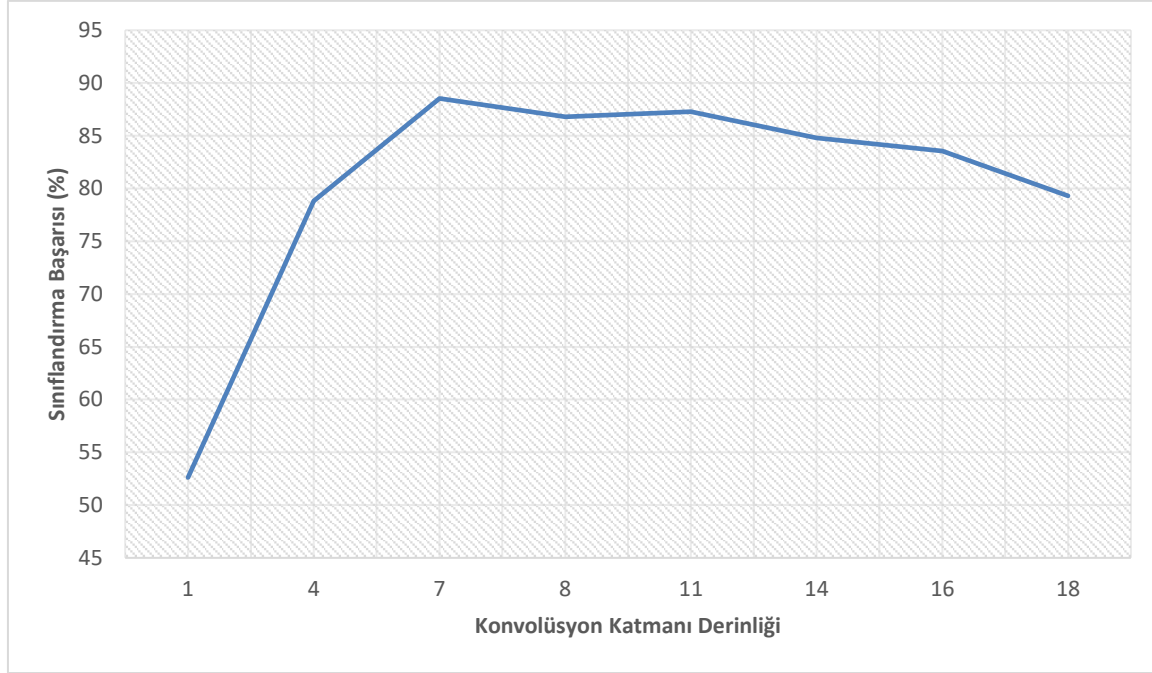


Şekil 4.8. QUT veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Özniteliklerin sınıflarla korelasyonu incelendikten sonra en iyi 1000 adet öznitelik ile 10 parçalı çapraz doğrulama yapılarak performans ölçümleri yapılmıştır. Elde edilen başarı oranları Çizelge 4.18’de ve Şekil 4.9’da verilmiştir.

Çizelge 4.18. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle QUT veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

Derinlik	Sınıflandırma Başarıları (%)
1	52,6185
4	78,803
7	88,5287
9	86,783
11	87,2818
14	84,788
16	83,5411
18	79,3017



Şekil 4.9. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle QUT veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

Çizelge 4.19. QUT veri setinde 7. katmandan çıkarılan öznitelikler kullanılarak elde edilen karmaşıklık matrisi

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	<-- sınıflandırma sonucu
15	5	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	a = cantherhines_dumerilii
1	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b = cantherhines_pardalis
0	1	24	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	c = caranx_melampygus
0	0	0	20	0	1	0	0	0	0	0	0	0	0	0	0	0	0	d = cheilinus_chlorourus
0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	e = cirrhilabrus_punctatus
0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	1	0	0	f = halichoeres_melanurus
0	0	0	0	0	0	19	0	0	0	0	0	1	0	0	0	1	0	g = hemigymnus_fasciatus
0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	2	0	0	h = leptojulius_cyanopleura
0	0	1	0	0	0	0	0	19	0	0	1	2	0	0	0	0	0	i = lethrinus_nebulosus
0	0	0	0	0	0	0	0	0	19	1	0	2	0	0	0	0	0	j = lutjanus_bohar
0	0	0	0	0	0	0	0	0	3	17	0	1	0	0	0	0	1	k = lutjanus_fulviflamma
0	0	0	0	0	0	0	0	0	0	0	20	1	0	0	0	0	0	l = lutjanus_sebae
1	0	1	0	0	0	0	0	2	0	1	0	16	0	0	1	0	0	m = monotaxis_grandoculis
0	1	0	0	0	1	0	0	0	0	0	0	0	24	0	0	0	0	n = oxymonacanthus_longirostris
0	0	0	0	0	0	0	0	0	0	0	0	1	0	22	0	0	0	o = pseudanthias_squamipinnis
0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	18	0	0	p = stethojulis_interrupta
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	19	1	q = thalassoma_lunare
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	19	r = thalassoma_lutescens

QUT alt veri seti üzerindeki sınıflandırma başarıları incelendiğinde (Çizelge 4.18, Şekil 4.9), bir önceki adımda elde edilen minimum ve maksimum korelasyon değerleri ile örtüşen sonuçlar alınmıştır. En yüksek değer, 7. katmandan alınan özniteliklerle elde edilmiştir. Katman derinliği artarken başlarda başarı yükselmiş ancak daha sonra düşmüştür. 7. katmandan çıkarılan özniteliklerle yapılan sınıflandırma sonunda elde edilen karmaşıklık matrisi Çizelge 4.19’da verilmiştir.

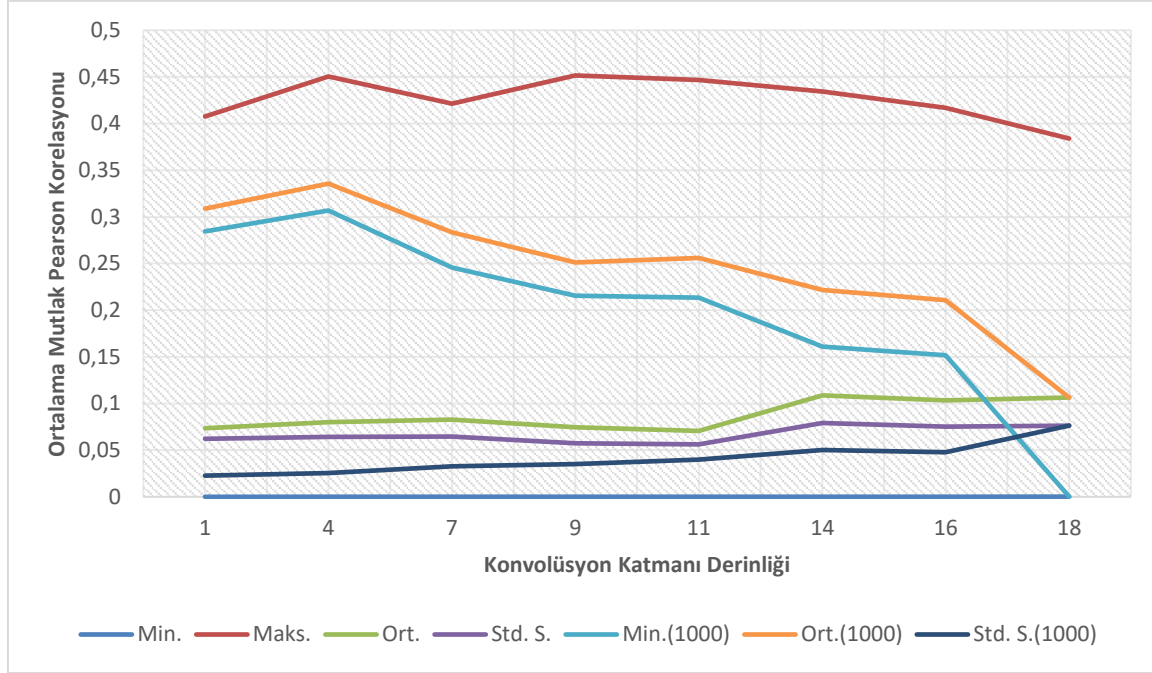
Ağın 7. katmanından çıkarılan özniteliklerle elde edilen %88,5287’lik başarı, tüm ağın öznitelik çıkarıcı olarak kullanıldığı durumda elde edilen %79,3017 değerinden belirgin olarak daha yüksektir. Bu değer, Mathur ve Goel tarafından aynı veri setinden elde edilen %84,92’luk başarının da üzerindedir (Mathur ve Goel, 2021).

4.4.5. AppleScabFDs elma görüntüleri veri seti

Aynı katman analizi çalışması, AppleScabFDs veri setinde de tekrarlandığında CNNFET ile elde edilen özniteliklerin sınıflarla aralarındaki ortalama mutlak Pearson korelasyonları Çizelge 4.20 ve Şekil 4.10’da verilmiştir.

Çizelge 4.20. AppleScabFDs veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Derinlik	Ortalama Mutlak Pearson Korelasyonları						
	Tümü				En İyi 1000 Öznitelik		
	Min.	Maks.	Ort.	Std. S.	Min.	Ort.	Std. S.
1	0,0000	0,4075	0,0736	0,0622	0,2843	0,3086	0,0227
4	0,0001	0,4504	0,0800	0,0643	0,3068	0,3356	0,0254
7	0,0001	0,4214	0,0826	0,0647	0,2458	0,2834	0,0327
9	0,0001	0,4515	0,0746	0,0573	0,2155	0,2513	0,0352
11	0,0001	0,4467	0,0706	0,0562	0,2135	0,2561	0,0398
14	0,0001	0,4344	0,1087	0,0791	0,1609	0,2215	0,0501
16	0,0001	0,4167	0,1034	0,0751	0,1518	0,2106	0,0476
18	0,0002	0,3839	0,1064	0,0763	0,0002	0,1064	0,0763



Şekil 4.10. AppleScabFDs veri setinde farklı derinliklerdeki katmanlardan elde edilen özniteliklerin sınıflarla ilişkileri

Çizelge 4.20 ve Şekil 4.10'a bakıldığında katman derinlikleri değiştikçe, elde edilen özniteliklerin sınıflar ile ilişkileri değişmektedir. Özniteliklerin tümü ele alındığında, en büyük korelasyon değeri, 9. katmanda elde edilen özniteliğe ait görünmektedir. Ortalama korelasyona bakıldığında, en yüksek ortalama 14. katmanda elde edilmiştir. 9. katmandan sonra derinlere ilerlendikçe, özniteliklerin sınıflarla maksimum korelasyonu belirgin bir şekilde düşmekteyken ortalama korelasyon 14. katmanda en yüksek değerindedir. Bu veriler ışığında, çok sayıda farklı nesne görüntüleri için eğitilmiş ağı katmanlarında sonlara yaklaşıldıkça, özniteliklerin ön eğitilmiş ağı eğitim verileri için daha da özelleştiği ve ağı eğitiminde kullanılmayan görüntüler için daha genel kaldığı sonucuna varılmıştır. Maksimum korelasyondaki düşüşe rağmen ortalama yükseliş, çok ayırt edici özniteliklerin ağı ortalarına doğru elde edilmiş olmasına rağmen, daha sonra elde edilen özniteliklerin, daha az ayırt edici ancak ilintili olanların sayıca daha fazla olduğunu göstermektedir.

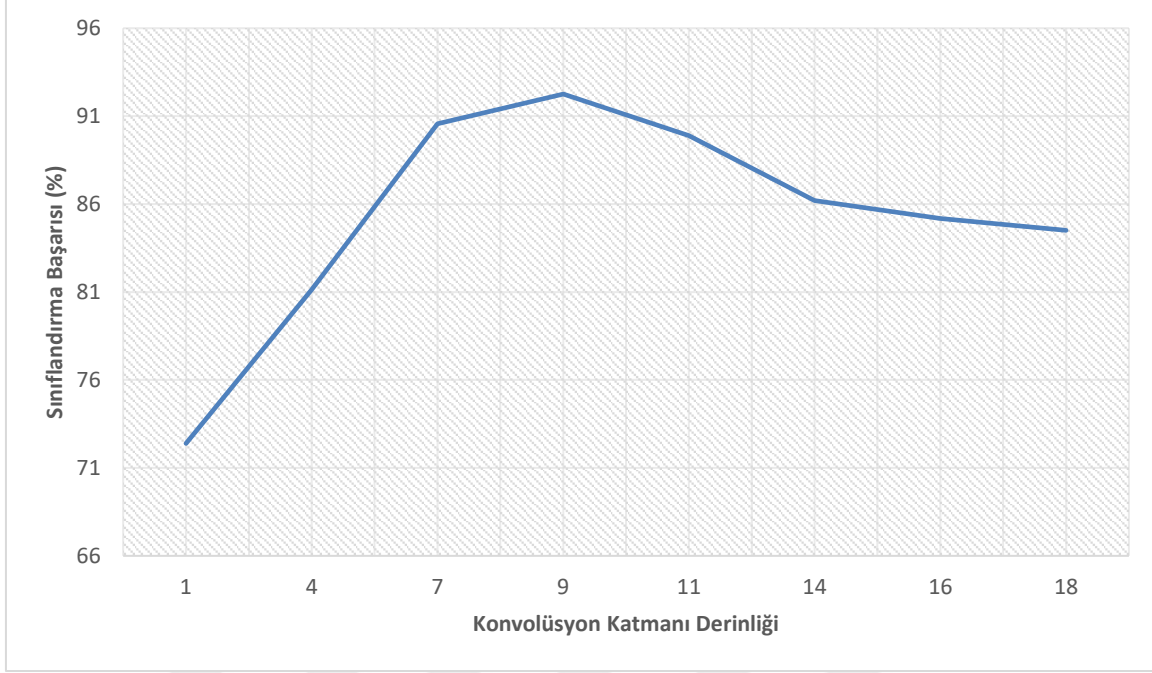
En iyi 1000 adet özniteliğe bakıldığında, en yüksek değerin 9. katmanda, en yüksek ortalama değerinse 4. katmanda olduğu görülmektedir. Standart sapmaların giderek artması, derinlere inildikçe öznitelik sayılarının azalmasına rağmen ayırt ediciliğin öznitelikler arasında dağıldığını göstermektedir.

Tablonun geneli, ayırt edici özniteliklerin ayırt etme nitelikleri bakımından ortalarında daha güçlü olduğunu ancak daha az sayıda özneliğin daha fazla ayırt edici olduğunu göstermektedir. Daha derin katmanlara inildikçe, özniteliklerin tekil olarak ayırt etme niteliğinin azaldığı ancak ayırt etme gücünün yine diğer özniteliklere dağıldığı sonucuna varılmaktadır.

Özniteliklerin sınıflarla korelasyonu incelendikten sonra en iyi 1000 adet öznitelik ile 10 parçalı çapraz doğrulama çalışması yürütülmüştür. Destek vektör makineleri, doğrusal çekirdek ile kullanılarak her katmandan alınan özniteliklerle AppleScabFDs veri seti üzerinde ayrı ayrı hastalıklı elma görüntüleri belirlenmiştir. Elde edilen başarı oranları Çizelge 4.21 ve Şekil 4.11’de verilmiştir.

Çizelge 4.21. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle AppleScabFDs veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen başarılar

Derinlik	Sınıflandırma Başarıları (%)
1	72,3906
4	81,1448
7	90,5724
9	92,2559
11	89,899
14	86,1953
16	85,1852
18	84,5118



Şekil 4.11. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle AppleScabFDs veri setinde 10 parçalı çapraz doğrulama yapılarak elde edilen sınıflandırma başarıları

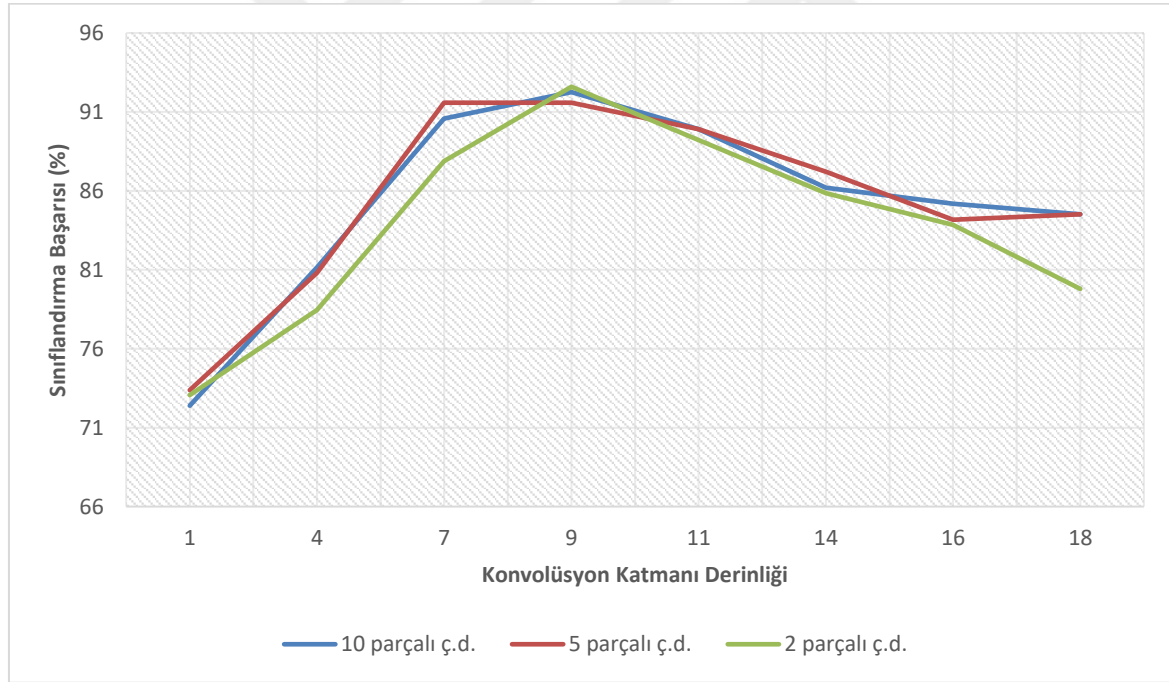
Değerler incelendiğinde bir önceki aşamada elde edilen korelasyonlara paralel olarak en iyi sınıflandırma başarısının 9. katmandan alınan özniteliklerle elde edildiği görülmektedir (Çizelge 4.21, Şekil 4.11). Sınıflandırma başarısı 9. katmana kadar artış gösterirken sonraki katmanlarda azalmaktadır.

Sınıflandırma çalışmaları, çapraz doğrulama aşamasında, parça sayısının sonuçları çok fazla etkilemediğinin gösterilmesi açısından, 5 ve 2 parçalı olarak tekrar edilmiştir. Elde edilen başarı sonuçları Çizelge 4.22 ve Şekil 4.12’de verilmiştir.

5 ve 2 parçalı çapraz doğrulama tekniği kullanıldığında, her parça ikilisinde, 10 parçalı çapraz doğrulama tekniğine göre eğitim setine daha az örnek düşerken test setine daha fazla örnek düşer. Sonuçlar bir önceki sonuçlarla birlikte değerlendirildiğinde, en yüksek başarı yine 9. katmandan alınan özniteliklerle elde edilmiştir. En yüksek başarı oranı olan bu başarı oranının elde edildiği çalışmada hesaplanan karmaşıklık matrisi Çizelge 4.23’de sunulmuştur.

Çizelge 4.22. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle AppleScabFDs veri setinde 5 ve 2 parçalı çapraz doğrulama yapılarak elde edilen başarılar

Derinlik	Sınıflandırma Başarıları (%)	
	5 parçalı çapraz doğrulama	2 parçalı çapraz doğrulama
1	73,374	73,064
4	80,8081	78,4512
7	91,5825	87,8788
9	91,5825	92,5926
11	89,899	89,2256
14	87,2054	85,8586
16	84,1751	83,8384
18	84,5118	79,798



Şekil 4.12. Farklı derinliklerdeki katmanlardan elde edilen özniteliklerle AppleScabFDs veri setinde 10, 5 ve 2 parçalı çapraz doğrulama yapılarak elde başarılar

Çizelge 4.23. AppleScab veri seti, 9. katmandan çıkarılan öznitelikler ve 2 parçalı çapraz doğrulama ile sınıflandırıldığında elde edilen karmaşıklık matrisi

a	b	<-- sınıflandırma sonucu
73	17	a = Sağlıklı
5	202	b = Hastalıklı

Literatürde bu veri seti üzerinde yapılan yüksek başarımlı çalışmaya dönüldüğünde, karşılaştırma yapılabilmesi için Çizelge 4.23 üzerinden hastalıklı elma görüntülerinin kesinlik, duyarlılık ve ikisinin harmonik ortalaması F1 değerleri sırasıyla %92,237, %97,585 ve %94,836 olarak hesaplanmıştır. Literatürdeki en başarılı sonuçlardan biri incelendiğinde (Nouaze ve Sikati, 2023), kesinlik, duyarlılık ve F1 değeri sırasıyla %94,8, %85,20 ve %89,75 olarak elde edilmiştir. Tez kapsamında daha önce bu veri seti üzerinde direkt olarak ağ çıkışından elde edilen özniteliklerle yapılan çalışmanın sonuçları düşük olsa da konvolüsyonel ağ üzerinde herhangi bir ek eğitim yapılmadan uygun katman arayışı ile başarının çok daha yüksek olabildiği görülmüştür.



5. SONUÇ VE ÖNERİLER

Temel amacı, literatüre disiplinler arası çalışmalarda kolaylıkla kullanılabilecek bir araç seti yazılması ve yazılan yazılımın etkinliğinin incelenmesi olan bu tez çalışmasının ilk aşamasında Fourier serileri ve parametreleri üzerinde çalışılmıştır. Fourier serileri yönteminin implementasyonu yapılarak BioMorph isimli bir araca modül olarak eklenmiştir.

Bir sonraki aşamada görüntü işleme konusunda daha modern bir yöntem olan konvolüsyonel yapay sinir ağları incelenmiştir. Konvolüsyonel ağların görüntülerde ayırt edici öznelikler yakalayabilme başarısı, bu ağları öznelik çıkarımı için oldukça kullanışlı araçlar haline getireceği öngörülerek, önceden farklı görüntüler üzerinde eğitilen konvolüsyonel ağ modellerinin farklı veriler üzerindeki başarısı araştırılmıştır. Bu aşamada CeNiN isiminde bir konvolüsyonel yapay sinir ağı kütüphanesi tasarlanmış ve implementasyonu yapılarak açık kaynak kodlu olarak yayınlanmıştır. Daha sonra bu kütüphane kullanılarak CNNFET isimli araç seti yazılmış ve yine açık kaynak kodlu olarak araştırmacıların kullanımına sunulmuştur.

Yazılan kütüphaneye ve araç setine BLAS desteğinin kazandırılmasının yanı sıra, yazılım tarafında mümkün olan her kısımda paralelleştirme uygulanarak işlemlerin hızlandırılması sağlanmıştır.

Yazılan kütüphanenin ve araç setinin etkinliğinin incelenebilmesi için farklı disiplinlerle alakalı farklı veri setleri incelenmiş ve bu veri setlerinin 6 tanesi çalışmaya dahil edilmiştir.

Fourier serileri yönteminde, sonuca doğrudan etki eden ilk parametre olan sinüs ve kosinüs terimlerinin sayısı (N) arttıkça, konturlarda yakalanan detaylar da beklendiği gibi artmaktadır. Ancak sayının fazlaca yüksek olması zaman açısından hesaplama maliyetini de arttırmaktadır.

Diğer bir değişken, konturların yeniden eldesi esnasında kontur başına istenen örnek sayısıdır. Bu sayı çok düşük seçildiğinde, konturlar Fourier serileri üzerinden düzgün örneklenememekte ve elde edilen yeni konturlarda önemli detaylar yok olmaktadır. Yaprak görüntüleri üzerinde yapılan çalışmalarda, en iyi parametrelerin seçimi için her iki parametrenin de belli iki değer arasındaki tüm kombinasyonları test edilmiş ve en iyi

parametre ikilisi belirlenmiştir. Buna göre bir yaprak görüntüsünün temsili için $N=7$ ve $M=12$ değerleri en yüksek başarının elde edilmesini sağlamıştır. Parametrelerin daha çok detay yakalamak için yükseltilmesinin başarıyı düşürdüğü de gözlenmiştir. Bunun sebebi, konturlardaki gürültüdür. Konturlar kusursuz olarak elde edilmediğinden, sınır olarak belirlenen çizgilerde gürültü olarak nitelendirilebilecek dalgalanmalar mevcut olacaktır. Bu da daha çok detay örneklendiğinde, başarının düşmesine sebep olmaktadır.

Fourier serileri için en iyi olarak nitelendirilebilecek bir parametre ikilisinden söz etmek zordur. Ancak bu şekilde bir en iyi parametre testi ile belli görüntü türleri için uygun parametreler tespit etmek mümkündür.

Tez çalışmasının devamında yazılan CeNiN kütüphanesi ve CNNFET araç seti ile yapılan testler, bir taraftan ön eğitilmiş ağların görüntü tanıma konusundaki başarısını bir kez daha teyit eden sonuçlar verirken öte yandan uygulamanın farklı disiplinlerde, farklı görüntülerin analizi için oldukça kullanışlı olabileceğini de göstermiştir. Geliştirilen araç seti kullanılarak çıkarılan ve işlenen öznetelik setleri, Weka üzerinde de test edilmiş, başarılı ve sorunsuz bir şekilde Weka ile de işlenebilmiştir.

Önce Fourier serileri yardımıyla yalnızca 10 türde 200 adet görüntüsü sınıflandırılan Flavia veri setinde en iyi sınıflandırma başarısı %98,5 olarak ölçülmüştür. Aynı veri setinin tamamında 32 türe ait 1907 adet görüntüde Fourier serileri yöntemi ile test tekrarlandığında başarı 75,197'ye düşmüştür. Veri setinin tamam üzerinde CNNFET ile öznetelik çıkarımı yapıldığında ise sınıflandırma başarısı %98,899 olarak ölçülmüştür. Tür sayısının iki kattan fazla artmasına rağmen, konvolüsyonel yapay sinir ağı ile elde edilen öznetelikler kullanılarak yapılan sınıflandırmanın başarısı Fourier serileri ile elde edilen başarıdan çok daha yüksek olmuştur.

Akciğer kanseri tomografileri üzerinde yürütülen testlerde de önceden eğitilmiş ağ, CNNFET yardımıyla tomografilerde iyi huylu, kötü huylu tümörleri ve tümörsüz tomografi görüntülerini ayırt eden öznetelikler yakalamayı başarmıştır. CNNFET yardımıyla elde edilen %95,989'luk sınıflandırma başarısı, araç setinin bu tür tıbbi görüntülerin analizlerinde de kullanılabileceğini göstermektedir.

102flowers veri seti üzerinde yapılan testlerde en az 100 görüntü içeren 23 sınıfa ait alt veri setinde %90,993'lük bir başarı elde edilmişse de sınıf sayısının 102'ye çıkarılması, başarıyı düşürmüştür. Kullanılan ön eğitilmiş ağı 1000 adet çıkışı olduğu göz önünde bulundurulduğunda 102 kategori için 1000 adet özniteliğin yetersiz olduğu sonucuna varmak mümkündür. Makine öğrenmesi alanında sınıflandırma problemlerinde, sınıf sayısı arttıkça sınıfların her biri için ayırt edici öznitelikler yakalamak zorlaşmaktadır. Sınıf sayısı arttıkça, daha çok sınıfı ayırmak için daha fazla ayırt edici öznitelige gereksinim duyulacaktır. Ayrıca sınıfların sayısının artması, birbirlerine benzeme olasılıklarını da artırır.

Bir başka balık veri seti olan QUT veri setinde, balık görüntüleri sabit olmayan, kontrolsüz ortamlarda farklı arka planlarda yer almaktadır. Buna rağmen CNNFET ile elde edilen öznitelikler kullanılarak, 18 türe ait balık görüntülerinin %79,3017'si doğru sınıflandırılmıştır.

CNNFET ile bir diğer veri seti olan ve yine kontrolsüz ortamda herhangi bir ön işlem yapılmadan çekilen elma görüntülerinden oluşan AppleScabFDs veri seti üzerinde yapılan çalışmada ise sağlıklı ve hastalıklı elma görüntüleri %84,512'lik bir başarı ile ayırt edilebilmiştir.

Çalışmaların devamında tüm veri setleri üzerinde, konvolüsyonel ağ katmanları ve korelasyonlar ile ilgili detaylı analizler gerçekleştirilmiş ve ön eğitilmiş bir yapay sinir ağının farklı tüm konvolüsyon katmanlarından ayrı öznitelikler alınmıştır. Alınan özniteliklerin sınıflarla korelasyonları hesaplanmış ve hangi katmanların daha ayırt edici öznitelikler çıkarabildiği incelenmiştir. Daha sonra ayrı ayrı her katmandan elde edilen özniteliklerle sınıflandırma işlemi gerçekleştirilmiş ve tüm veri setlerinde ağın tamamı öznitelik çıkarıcı olarak kullanıldığında elde edilen başarıdan daha yüksek başarılar elde edilebilmiştir.

Ön eğitilmiş konvolüsyonel yapay sinir ağının tamamının öznitelik çıkarıcı olarak kullanılması yerine, CNNFET yardımıyla ağdaki konvolüsyonel katmanların çıkışlarının sınıflarla ilişkileri incelenerek bir ara katmanlardan çıkış alınması ile Flavia yaprak görüntüleri veri setinde başarı, %98,8988'den 14. katmanda %99,4232'ye, IQ-OTH_NCCD tomografi görüntüleri veri setinde %95,9891'den 11. katmanda 98,5415'e, 102flowers çiçek görüntüleri veri setinde %90,9926'dan 14. katmanda %93,3333'e, QUT balık görüntüleri

veri setinde %79,3017'den 7. katmanda %88,5287'ye ve AppleScabFDs elma görüntüleri veri setindey %84,5118'den %92,2559'a yükseltilmiştir.

Böylece yapılan katman ve öznitelik analiz çalışmaları ile ön eğitilmiş ağların, farklı katmanlarda, kullanılacakları veri setleri için daha ayırt edici öznitelikler yakalayabileceği gösterilmiştir. Bir ön eğitilmiş ağın direkt olarak çıkışlarının veya son katmanlara yakın katman çıkışlarının kullanılması, ağın eğitildiği görüntüler için daha da özelleşmiş sonuçlar vereceğinden, elde edilecek özniteliklerin başarısını düşürebilir. Dolayısıyla bir ön eğitilmiş ağ kullanılırken, veri seti için en uygun katmanın, öznitelikler ile sınıflar arasındaki korelasyonlar incelenerek ve/veya ağın farklı katmanlarının veri setleri üzerindeki başarısı incelenerek belirlenmesi, ağ ile veri seti üzerinde mümkün olan en yüksek başarının yakalanmasını sağlayacaktır.

Aynı çalışmalarla, konvolüsyonel yapay sinir ağlarının başlangıçta öğrendiği filtreler ile yüksek çeşitlilikli öznitelikler elde edilirken, ilerleyen katmanlarda özniteliklerin eğitim veri için daha da özelleştiği sonucuna varılmıştır. Konvolüsyonel yapay sinir ağlarının ilk katmanlarından itibaren öznitelik çıkarımı konusunda başarılı sonuçlar alınırken, aynı zamanda ilerleyen katmanların öznitelik indirgeme aracı görevi de gördüğü de göz önünde bulundurulmalıdır.

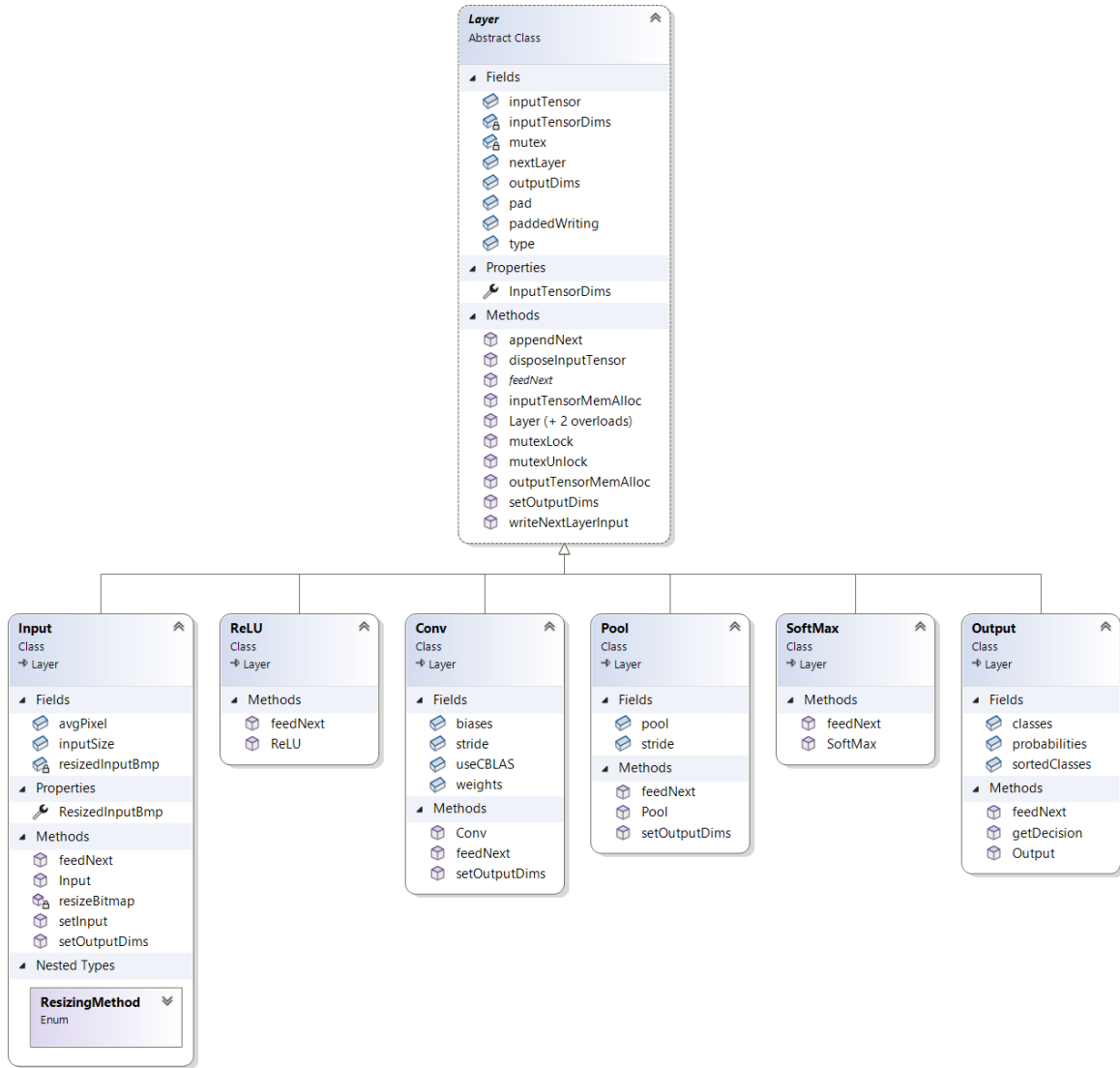
Tez çalışması kapsamında, testlerle etkinliği gösterilen CeNiN ve CNNFET isimli bir kütüphane ve bir Windows uygulaması yazılmıştır. Kütüphane ve araç seti açık kaynak kodlu ve geliştirmeye açık yazılımlardır. Araç setine farklı sınıflandırıcılar, farklı öznitelik seçim yöntemleri, farklı dosya destekleri eklenmesi mümkündür.

Konvolüsyonel yapay sinir ağlarının sınıflandırma problemleri için ayırt edici öznitelikler yakalama becerilerinin, bir benzerlik ölçüm aracı olarak da kullanılabilmesini mümkün hale getirebileceği düşünülmektedir. İleriki çalışmalarda, CNNFET ile çıkarılan özniteliklerin sınıf vektörleriyle korelasyonları ve farklı sınıflandırıcılarda sağlayacakları başarı daha fazla veri seti üzerinden incelenerek, bir veri setinde yer alan farklı sınıfların benzerliği ile ilgili değerlendirmeler yapıp yapılamayacağının da incelenmesi planlanmaktadır. CNNFET, buna benzer pek çok çalışma için kullanışlı bir araç seti olacaktır.

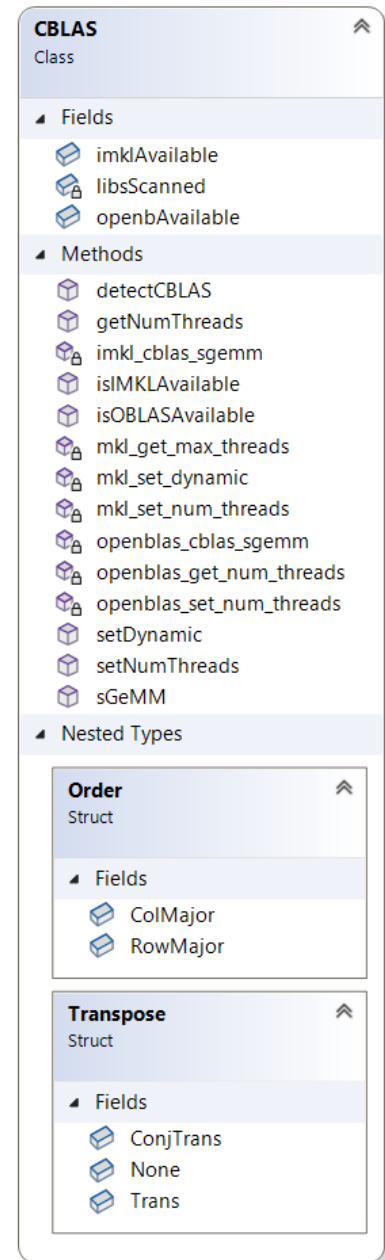
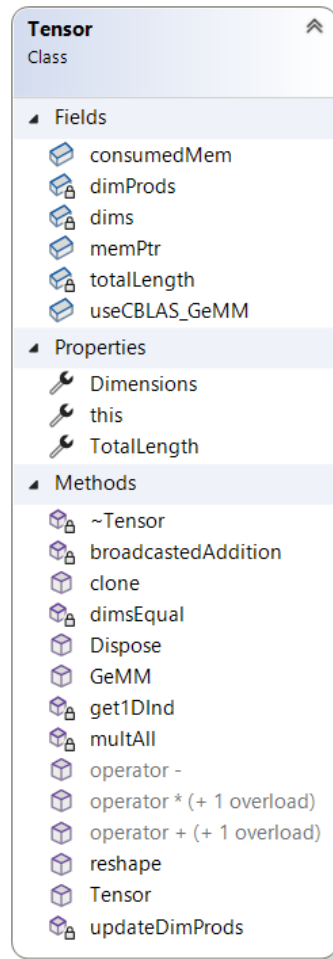
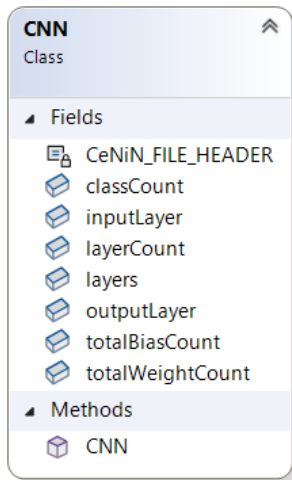
Geliştirilen araç seti farklı disiplinlerde çalışan araştırmacıların konvolüsyonel yapay sinir ağlarını, kendi alanları dışında çok fazla teknik bilgi veya kodlama becerisi gerekmeksizin çalışmalarında kullanabilmelerine imkân sağlayacaktır.



EK A: CeNiN Kütüphanesi – Katmanların Sınıf Diyagramları



EK B: CeNiN Kütüphanesi – Diğer Sınıfların Sınıf Diyagramları



EK C: CeNiN Kütüphanesi – Kaynak Kodlar

```

unsafe public class CNN
{
    private const string CeNiN_FILE_HEADER = "CeNiN NEURAL NETWORK FILE";

    public int layerCount;
    public int classCount;
    public int totalWeightCount;
    public int totalBiasCount;

    public Layer[] layers;
    public Input inputLayer;
    public Output outputLayer;

    public CNN(string path)
    {
        bool useCBLAS = CBLAS.detectCBLAS();

        FileStream f = null;
        BinaryReader br = null;
        try
        {
            f = new FileStream(path, FileMode.Open);
            br = new BinaryReader(f, Encoding.ASCII);
            char[] c = br.ReadChars(25);
            if (!(new string(c)).Equals(CeNiN_FILE_HEADER))
                throw new Exception("Invalid file header!");

            layerCount = br.ReadInt32();
            int[] inputSize = new int[3];
            for (int i = 0; i < 3; i++)
                inputSize[i] = br.ReadInt32();

            inputLayer = new Input(inputSize);

            for (int i = 0; i < 3; i++)
                inputLayer.avgPixel[i] = br.ReadSingle();

            inputLayer.setOutputDims();

            Layer layerChain = inputLayer;
            Layer currentLayer = inputLayer;

            totalWeightCount = 0;
            totalBiasCount = 0;
        }
        finally
        {
            if (f != null) f.Close();
            if (br != null) br.Close();
        }
    }
}

```

```

List<Layer> layerList = new List<Layer>();
layerList.Add(currentLayer);

bool endOfFile = false;
while (!endOfFile)
{
    string layerT = br.ReadString();
    if (layerT.Equals("conv"))
    {
        int[] pad = new int[4];
        for (int i = 0; i < 4; i++)
            pad[i] = br.ReadByte();

        int[] inputTensorDims = currentLayer.outputDims;
        Conv cLayer = new Conv(inputTensorDims, pad);
        cLayer.useCBLAS = useCBLAS;

        int[] dims = new int[4];
        for (int i = 0; i < 4; i++)
            dims[i] = br.ReadInt32();

        for (int i = 0; i < 2; i++)
            cLayer.stride[i] = br.ReadByte();

        cLayer.weights = new Tensor(dims);
        for (int i = 0; i < cLayer.weights.TotalLength; i++)
            cLayer.weights.memPtr[i] = br.ReadSingle();
        totalWeightCount += cLayer.weights.TotalLength;

        cLayer.biases = new Tensor(new int[] { dims[3] });
        for (int i = 0; i < cLayer.biases.TotalLength; i++)
            cLayer.biases.memPtr[i] = br.ReadSingle();
        totalBiasCount += cLayer.biases.TotalLength;

        cLayer.setOutputDims();

        currentLayer = cLayer;
    }
    else if (layerT.Equals("relu"))
    {
        ReLU rLayer = new ReLU(currentLayer.outputDims);
        rLayer.setOutputDims();
        currentLayer = rLayer;
    }
    else if (layerT.Equals("pool"))
    {
        int[] pad = new int[4];
        for (int i = 0; i < 4; i++)
            pad[i] = br.ReadByte();

```

```

        Pool pLayer = new Pool(currentLayer.outputDims, pad);

        for (int i = 0; i < 2; i++)
            pLayer.pool[i] = br.ReadByte();

        for (int i = 0; i < 2; i++)
            pLayer.stride[i] = br.ReadByte();

        pLayer.setOutputDims();
        currentLayer = pLayer;
    }
    else if (layerT.Equals("softmax"))
    {
        classCount = br.ReadInt32();
        string[] classes = new string[classCount];
        for (int i = 0; i < classCount; i++)
            classes[i] = br.ReadString();

        SoftMax smLayer = new SoftMax(currentLayer.outputDims);
        currentLayer.appendNext(smLayer);
        outputLayer = new Output(smLayer.InputTensorDims,
            classes);
        smLayer.appendNext(outputLayer);
        layerList.Add(smLayer);
        layerList.Add(outputLayer);
        continue;
    }
    else if (layerT.Equals("EOF"))
    {
        endOfFile = true;
        continue;
    }
    else
        throw new Exception("The following layer is not
            implemented: " + layerT);

    layerList.Add(currentLayer);
    layerChain.appendNext(currentLayer);
    layerChain = layerChain.nextLayer;
}
layers = layerList.ToArray();
}
catch (Exception)
{
}
finally
{
    if (br != null) br.Close();
}

```

```

        if (f != null) f.Close();
    }
}

public unsafe class CBLAS
{
    private static bool libsScanned = false;

    public static bool imklAvailable = false;
    public static bool openbAvailable = false;

    [DllImport("openblas.dll", EntryPoint = "cblas_sgemm", CallingConvention = CallingConvention.StdCall)]
    private static extern void openblas_cblas_sgemm(int order, int transA, int transB, int m, int n, int k, float alpha, float* A, int lda, float* B, int ldb, float beta, float* C, int ldc);

    [DllImport("mkl_rt.dll", EntryPoint = "cblas_sgemm", CallingConvention = CallingConvention.Cdecl)]
    private static extern void imkl_cblas_sgemm(int order, int transA, int transB, int m, int n, int k, float alpha, float* A, int lda, float* B, int ldb, float beta, float* C, int ldc);

    public static bool detectCBLAS()
    {
        if (!libsScanned)
        {
            if (isOBLASAvailable())
                openbAvailable = true;
            if (isIMKLAvaliable())
                imklAvailable = true;

            libsScanned = true;
        }
        return openbAvailable || imklAvailable;
    }

    public static void sGeMM(int order, int transA, int transB, int m, int n, int k, float alpha, float* A, int lda, float* B, int ldb, float beta, float* C, int ldc)
    {
        if (!libsScanned)
            detectCBLAS();
        if (openbAvailable)
            openblas_cblas_sgemm(order, transA, transB, m, n, k, alpha, A, lda, B, ldb, beta, C, ldc);
        else if (imklAvailable)
            imkl_cblas_sgemm(order, transA, transB, m, n, k, alpha, A, lda,

```

```

        B, ldb, beta, C, ldc);
    }

[DllImport("openblas.dll", CallingConvention = CallingConvention.Cdecl)]
private static extern void openblas_set_num_threads(int n);

[DllImport("openblas.dll", CallingConvention = CallingConvention.Cdecl)]
private static extern int openblas_get_num_threads();

[DllImport("mkl_rt.dll", CallingConvention = CallingConvention.Cdecl)]
private static extern void mkl_set_dynamic(int* d);

[DllImport("mkl_rt.dll", CallingConvention = CallingConvention.Cdecl)]
private static extern void mkl_set_num_threads(int* n);

[DllImport("mkl_rt.dll", CallingConvention = CallingConvention.Cdecl)]
private static extern int mkl_get_max_threads();

public static void setNumThreads(int n)
{
    int i = 0;

    if (openbAvailable)
        openblas_set_num_threads(n);
    else if (imklAvailable)
    {
        mkl_set_dynamic(&i);
        mkl_set_num_threads(&n);
    }
}

public static void setDynamic()
{
    int i = 1;
    if (imklAvailable)
        mkl_set_dynamic(&i);
}

public static int getNumThreads()
{
    if (openbAvailable)
        return openblas_get_num_threads();
    else if (imklAvailable)
        return mkl_get_max_threads();
    return -1;
}

public static bool isOBLASAvailable()
{

```

```

        string[] libs = { "openblas.dll", "libquadmath-0.dll",
        "libgfortran-3.dll", "libgcc_s_seh-1.dll" };
        for (int i = 0; i < libs.Length; i++)
            if (!File.Exists(libs[i]))
                return false;

        return true;
    }

    public static bool isIMKLAvaliable()
    {
        string[] libs = { "mkl_rt.dll", "mkl_intel_thread.dll",
        "mkl_core.dll", "libiomp5md.dll", "mkl_def.dll", "mkl_avx.dll",
        "mkl_avx2.dll", "mkl_avx512.dll", "mkl_mc.dll", "mkl_mc3.dll" };
        for (int i = 0; i < libs.Length; i++)
            if (!File.Exists(libs[i]))
                return false;

        return true;
    }

    public struct Order
    {
        public static int RowMajor = 101;
        public static int ColMajor = 102;
    }

    public struct Transpose
    {
        public static int None = 111;
        public static int Trans = 112;
        public static int ConjTrans = 113;
    }
}

public unsafe class Tensor
{
    private int totalLength;
    private int[] dims;

    public float* memPtr;
    public int consumedMem;

    private int[] dimProds;

    public static bool useCBLAS_GeMM = false;

    public int TotalLength
    {

```

```

        get
        {
            return totalLength;
        }
    }

    public int[] Dimensions
    {
        get
        {
            return dims;
        }
    }

    public Tensor(int[] dims)
    {
        totalLength = multAll(dims);

        this.dims = (int[])dims.Clone();
        updateDimProds();

        consumedMem = totalLength * sizeof(float);
        memPtr = (float*)Marshal.AllocHGlobal(consumedMem);
        for (int i = 0; i < totalLength; i++)
            memPtr[i] = 0.0f;
    }

    private void updateDimProds()
    {
        dimProds = new int[dims.Length];
        dimProds[0] = 1;
        for (int i = 1; i < dims.Length; i++)
            dimProds[i] = dimProds[i - 1] * dims[i - 1];
    }

    public float this[params int[] indexes]
    {
        get
        {
            int ind = get1DInd(indexes);
            return memPtr[ind];
        }
        set
        {
            int ind = get1DInd(indexes);
            memPtr[ind] = value;
        }
    }
}

```

```

private int get1DInd(int[] indexes)
{
    int ind = indexes[0];
    for (int i = 1; i < indexes.Length; i++)
        ind += dimProds[i] * indexes[i];

    if (ind >= totalLength)
        throw new IndexOutOfRangeException();
    return ind;
}

private static int multAll(int[] array)
{
    int mul = 1;
    for (int i = 0; i < array.Length; i++)
        mul *= array[i];
    return mul;
}

public bool reshape(int[] newDims)
{
    if (multAll(dims) != multAll(newDims))
        return false;
    dims = (int[])newDims.Clone();
    updateDimProds();
    return true;
}

private static bool dimsEqual(Tensor t1, Tensor t2)
{
    if (t1.dims.Length != t2.dims.Length)
        return false;
    for (int i = 0; i < t1.dims.Length; i++)
        if (t1.dims[i] != t2.dims[i])
            return false;
    return true;
}

public Tensor clone()
{
    Tensor t = new Tensor(dims);
    for (int i = 0; i < totalLength; i++)
        t.memPtr[i] = memPtr[i];
    return t;
}

public static Tensor operator *(Tensor t1, float f)
{
    Tensor t = new Tensor(t1.dims);

```

```

        for (int i = 0; i < t.totalLength; i++)
            t.memPtr[i] = t1.memPtr[i] * f;
        return t;
    }

    public static Tensor operator +(Tensor t1, float f)
    {
        Tensor t = new Tensor(t1.dims);
        for (int i = 0; i < t.totalLength; i++)
            t.memPtr[i] = t1.memPtr[i] + f;
        return t;
    }

    public static Tensor operator -(Tensor t1, float f)
    {
        return t1 + (-f);
    }

    public static Tensor operator +(Tensor t1, Tensor t2)
    {
        if (
            t1.dims.Length == 2 && t2.dims.Length == 2 &&
            (
                (t1.dims[0] == 1 && t2.dims[1] == 1) ||
                (t1.dims[1] == 1 && t2.dims[0] == 1)
            )
        )
            return broadcastedAddition(t1, t2);

        if (!dimsEqual(t1, t2))
            return null;

        Tensor t = new Tensor(t1.dims);

        for (int i = 0; i < t1.totalLength; i++)
            t.memPtr[i] = t1.memPtr[i] + t2.memPtr[i];

        return t;
    }

    public void GeMM(Tensor A, Tensor B, float alpha, float beta)
    {
        int m = A.Dimensions[0];
        int n = B.Dimensions[1];
        int k = A.Dimensions[1];
        int lda = m, ldb = k, ldc = m;

        CBLAS.sGeMM(
            CBLAS.Order.ColMajor,

```

```

        CBLAS.Transpose.None,
        CBLAS.Transpose.None,
        m, n, k,
        alpha, A.memPtr, lda,
        B.memPtr, ldb,
        beta, this.memPtr, ldc);
    }

    public static Tensor operator *(Tensor t1, Tensor t2)
    {
        if (t1.dims.Length != 2 || t2.dims.Length != 2 || t1.dims[1] !=
            t2.dims[0])
            return null;

        Tensor t = new Tensor(new int[] { t1.dims[0], t2.dims[1] });

        if (useCBLAS_GeMM)
        {
            t.GeMM(t1, t2, 1, 0);
            return t;
        }

        float sum;
        int[] ind1 = new int[] { 0, 0 };
        int[] ind2 = new int[] { 0, 0 };
        int[] ind3 = new int[] { 0, 0 };

        for (int i = 0; i < t1.dims[0]; i++)
        {
            ind1[0] = i;
            ind3[0] = i;
            for (int k = 0; k < t2.dims[1]; k++)
            {
                ind2[1] = k;
                ind3[1] = k;
                sum = 0;
                for (int j = 0; j < t1.dims[1]; j++)
                {
                    ind1[1] = j;
                    ind2[0] = j;
                    sum += t1[ind1] * t2[ind2];
                }
                t[ind3] = sum;
            }
        }

        return t;
    }
}

```

```

private static Tensor broadcastedAddition(Tensor t1, Tensor t2)
{
    int dim1 = t1.totalLength;
    int dim2 = t2.totalLength;

    Tensor t = new Tensor(new int[] { dim1, dim2 });
    int[] ind = new int[] { 0, 0 };
    for (ind[0] = 0; ind[0] < dim1; ind[0]++)
        for (ind[1] = 0; ind[1] < dim2; ind[1]++)
            t[ind] = t1.memPtr[ind[0]] + t2.memPtr[ind[1]];

    return t;
}

public void Dispose()
{
    if (memPtr != null)
    {
        Marshal.FreeHGlobal((IntPtr)memPtr);
        memPtr = null;
    }
}

~Tensor()
{
    Dispose();
}
}

public abstract class Layer
{
    private Mutex mutex;

    public string type;

    private int[] inputTensorDims;
    public int[] InputTensorDims
    {
        get
        {
            return inputTensorDims;
        }
    }

    public bool paddedWriting;
    public int[] pad;

    public Tensor inputTensor = null;
    public Layer nextLayer;
}

```

```

public Layer()
{
    mutex = new Mutex();
}

public void mutexLock()
{
    mutex.WaitOne();
}

public void mutexUnlock()
{
    mutex.ReleaseMutex();
}

public void writeNextLayerInput(int[] indexes, float value)
{
    if (nextLayer.paddedWriting)
    {
        int[] nInd = (int[])indexes.Clone();
        nInd[0] += nextLayer.pad[0];
        nInd[1] += nextLayer.pad[2];
        nextLayer.inputTensor[nInd] = value;
    }
    else
        nextLayer.inputTensor[indexes] = value;
}

public Layer(int[] inputTensorDims) : this()
{
    paddedWriting = false;
    this.inputTensorDims = (int[])inputTensorDims.Clone();
}

public Layer(int[] inputTensorDims, int[] pad) : this()
{
    this.pad = (int[])pad.Clone();
    if (pad[0] > 0 || pad[2] > 0)
        paddedWriting = true;
    else
        paddedWriting = false;

    this.inputTensorDims = (int[])inputTensorDims.Clone();
    this.inputTensorDims[0] += pad[0] + pad[1];
    this.inputTensorDims[1] += pad[2] + pad[3];
}

public int[] outputDims;

public void setOutputDims()

```

```

{
    outputDims = (int[])inputTensorDims.Clone();
}

public abstract void feedNext();

public void inputTensorMemAlloc()
{
    mutexLock();
    inputTensor = new Tensor(inputTensorDims);
}

public void outputTensorMemAlloc()
{
    nextLayer.inputTensorMemAlloc();
}

public void disposeInputTensor()
{
    inputTensor.Dispose();
    inputTensor = null;
    mutexUnlock();
}

public void appendNext(Layer nextLayer)
{
    this.nextLayer = nextLayer;
}
}

public class Conv : Layer
{
    public int[] stride;

    public Tensor weights;
    public Tensor biases;

    public bool useCBLAS = false;

    public Conv(int[] inputTensorDims, int[] pad) : base(inputTensorDims,
pad)
    {
        type = "Convolution";
        stride = new int[2];
    }

    public new void setOutputDims()
    {
        int newHeight = (int)Math.Floor((double)(InputTensorDims[0] -

```

```

weights.Dimensions[0]) / stride[0]) + 1;
int newWidth = (int)Math.Floor((double)(InputTensorDims[1] -
weights.Dimensions[1]) / stride[1]) + 1;

outputDims = new int[] { newHeight, newWidth,
weights.Dimensions[3] };
}

public unsafe override void feedNext()
{
    outputTensorMemAlloc();

    int inputHeight = InputTensorDims[0];
    int inputWidth = InputTensorDims[1];
    int filterHeight = weights.Dimensions[0];
    int filterWidth = weights.Dimensions[1];
    int channelCount = weights.Dimensions[2];
    int filterCount = weights.Dimensions[3];

    int mCountH = inputHeight - filterHeight + 1;
    int mCountW = inputWidth - filterWidth + 1;
    Tensor possibleH = new Tensor(new int[] { outputDims[0], 1 });
    int j = 0;
    for (int i = 0; i < mCountH; i += stride[0])
        possibleH.memPtr[j++] = i;

    Tensor possibleW = new Tensor(new int[] { 1, outputDims[1] });
    j = 0;
    for (int i = 0; i < mCountW; i += stride[1])
        possibleW.memPtr[j++] = i;

    Tensor startingIndexes = possibleW + possibleH * inputWidth;
    possibleH.Dispose();
    possibleW.Dispose();

    possibleH = new Tensor(new int[] { filterHeight, 1 });
    for (int i = 0; i < filterHeight; i++)
        possibleH.memPtr[i] = i;

    possibleW = new Tensor(new int[] { 1, filterWidth });
    for (int i = 0; i < filterWidth; i++)
        possibleW.memPtr[i] = i;

    Tensor offsets = possibleW + possibleH * inputWidth;
    possibleH.Dispose();
    possibleW.Dispose();

    startingIndexes.reshape(new int[] { startingIndexes.TotalLength,

```

```

1});
offsets.reshape(new int[] { 1, offsets.TotalLength });
Tensor allIndexes = startingIndexes + offsets;
startingIndexes.Dispose();
offsets.Dispose();

int outputH_W = outputDims[0] * outputDims[1];

Tensor allInOne = new Tensor(new int[] { outputH_W, filterHeight *
filterWidth * channelCount });

int h_W = inputHeight * inputWidth;
int fH_fW = filterHeight * filterWidth;
int h_w_fH_fW = h_W * fH_fW;
int tmp;

int aiInd, aioInd;
for (int ch = 0; ch < channelCount; ch++)
{
    int fH_fW_ch = fH_fW * ch;
    int h_W_ch = h_W * ch;
    for (int m = 0; m < fH_fW; m++)
    {
        aiInd = m * outputH_W;
        aioInd = (fH_fW_ch + m) * outputH_W;
        for (int k = 0; k < outputH_W; k++)
        {
            tmp = (int)allIndexes.memPtr[aiInd++] + h_W_ch;
            allInOne.memPtr[aioInd++] = inputTensor.memPtr[tmp];
        }
    }
}

allIndexes.Dispose();

nextLayer.inputTensor.reshape(new int[] { allInOne.Dimensions[0],
filterCount });

if (useCBLAS)
{
    weights.reshape(new int[] { filterHeight * filterWidth *
channelCount, filterCount });

    int q = 0;
    for (int i = 0; i < nextLayer.inputTensor.Dimensions[1]; i++)
    {
        float f = biases.memPtr[i];
        for (int p = 0; p < nextLayer.inputTensor.Dimensions[0];
p++)

```

```

        nextLayer.inputTensor.memPtr[q++] = f;
    }

    nextLayer.inputTensor.GeMM(allInOne, weights, 1.0f, 1.0f);

    weights.reshape(new int[] { filterHeight, filterWidth,
        channelCount, filterCount });
}
else
{
    int x = allInOne.Dimensions[1];
    int y = filterCount;
    int z = allInOne.Dimensions[0];

    ParallelOptions po = new ParallelOptions();
    po.MaxDegreeOfParallelism = Environment.ProcessorCount;

    Parallel.For(0, y, po, f =>
    {
        int aioInd_;
        int outputInd_;
        int weightsInd_;

        for (int g = 0; g < z; g++)
        {
            float sum = 0;
            for (int h = 0; h < x; h++)
            {
                aioInd_ = h * z + g;
                weightsInd_ = f * x + h;
                sum += weights.memPtr[weightsInd_] *
                    allInOne.memPtr[aioInd_];
            }
            outputInd_ = f * z + g;
            nextLayer.inputTensor.memPtr[outputInd_] = sum +
                biases.memPtr[f];
        }
    });
}

nextLayer.inputTensor.reshape(outputDims);

allInOne.Dispose();

disposeInputTensor();
}

public unsafe class Input : Layer

```

```

{
    public int[] inputSize;
    public float[] avgPixel;

    public enum ResizingMethod
    {
        Stretch,
        ZeroPad
    }

    private Bitmap resizedInputBmp;
    public Bitmap ResizedInputBmp
    {
        get
        {
            return resizedInputBmp;
        }
    }

    public Input(int[] inputTensorDims) : base(new int[] { 0, 0, 0 })
    {
        type = "Input";
        inputSize = (int[])inputTensorDims.Clone();
        avgPixel = new float[3];
    }

    public new void setOutputDims()
    {
        outputDims = (int[])inputSize.Clone();
    }

    private Bitmap resizeBitmap(Bitmap b, ResizingMethod resizingMethod)
    {
        Bitmap resizedBmp = new Bitmap(inputSize[1], inputSize[0],
            PixelFormat.Format24bppRgb);
        Graphics gr = Graphics.FromImage(resizedBmp);

        if (resizingMethod == ResizingMethod.Stretch)
            gr.DrawImage(b, 0, 0, inputSize[1], inputSize[0]);
        else
        {
            float inputAspRatio = inputSize[0] / inputSize[1];

            int newHeight, newWidth;
            float multiplier = (float)b.Width / b.Height;
            if (multiplier > inputAspRatio)
            {
                multiplier = inputAspRatio / multiplier;
                newWidth = inputSize[1];
            }
        }
    }
}

```

```

        newHeight = (int)(newWidth * multiplier);
    }
    else
    {
        newHeight = inputSize[0];
        newWidth = (int)(newHeight * multiplier);
    }

    gr.DrawImage(b, (inputSize[1] - newWidth) / 2.0f, (inputSize[0]
        - newHeight) / 2.0f, newWidth, newHeight);
}

gr.Dispose();

return resizedBmp;
}

public void setInput(Bitmap input, ResizingMethod resizingMethod)
{
    mutexLock();

    outputTensorMemAlloc();

    Bitmap iBitmap = (Bitmap)input.Clone();
    resizedInputBmp = resizeBitmap(iBitmap, resizingMethod);
    iBitmap.Dispose();
}

public override void feedNext()
{
    BitmapData bmpData = resizedInputBmp.LockBits(new Rectangle(0, 0,
        inputSize[1], inputSize[0]), ImageLockMode.ReadOnly,
        resizedInputBmp.PixelFormat);

    int stride = bmpData.Stride;
    int emptyBytesCount = stride - bmpData.Width * 3;
    int rowLengthWithoutEB = stride - emptyBytesCount;
    byte* dataPtr = (byte*)bmpData.Scan0.ToPointer();
    int byteCount = stride * bmpData.Height;

    int i = 0;
    int pControl = 0;
    int b, g, r;
    int[] ind = new int[] { 0, 0, 0 };
    while (i < byteCount)
    {
        b = *dataPtr;
        dataPtr++;
        g = *dataPtr;

```

```

        dataPtr++;
        r = *dataPtr;
        dataPtr++;

        ind[2] = 0;
        writeNextLayerInput(ind, r - avgPixel[0]);
        ind[2] = 1;
        writeNextLayerInput(ind, g - avgPixel[1]);
        ind[2] = 2;
        writeNextLayerInput(ind, b - avgPixel[2]);

        ind[1] += 1;

        pControl += 3;
        i += 3;
        if (pControl == rowLengthWithoutEB)
        {
            pControl = 0;
            dataPtr += emptyBytesCount;

            ind[1] = 0;
            ind[0] += 1;
        }
    }

    resizedInputBmp.UnlockBits(bmpData);
    resizedInputBmp.Dispose();

    mutexUnlock();
}

public class Output : Layer
{
    public string[] classes;

    public string[] sortedClasses;
    public float[] probabilities;

    public Output(int[] inputTensorDims, string[] classes)
    : base(inputTensorDims)
    {
        type = "Output";
        this.classes = classes;
        probabilities = new float[inputTensorDims[2]];
        sortedClasses = new string[inputTensorDims[2]];
    }

    public unsafe string getDecision()

```

```

{
    if (inputTensor.memPtr != null)
    {
        Array.Copy(classes, sortedClasses, classes.Length);
        Marshal.Copy((IntPtr)inputTensor.memPtr, probabilities, 0,
            classes.Length);
        Array.Sort(probabilities, sortedClasses);
        Array.Reverse(probabilities);
        Array.Reverse(sortedClasses);
        disposeInputTensor();
    }
    return sortedClasses[0];
}

public override void feedNext()
{
    throw new NotImplementedException();
}
}

public class Pool : Layer
{
    public int[] pool;
    public int[] stride;

    public Pool(int[] inputTensorDims, int[] pad)
        : base(inputTensorDims, pad)
    {
        type = "Pool";
        pool = new int[2];
        stride = new int[2];
    }

    public new void setOutputDims()
    {
        outputDims = new int[3] {
            (int)Math.Floor((double)InputTensorDims[0] / stride[0]),
            (int)Math.Floor((double)InputTensorDims[1] / stride[1]),
            InputTensorDims[2]
        };
    }

    public override void feedNext()
    {
        outputTensorMemAlloc();

        int inputHeight = InputTensorDims[0];
        int inputWidth = InputTensorDims[1];
        int channelCount = InputTensorDims[2];
    }
}

```

```

int poolHeight = pool[0];
int poolWidth = pool[1];

int[] inputInd = new int[] { 0, 0, 0 };
int[] outputInd = new int[] { 0, 0, 0 };

float max;
for (inputInd[2] = 0; inputInd[2] < channelCount; inputInd[2]++)
{
    outputInd[2] = inputInd[2];
    for (int i = 0; i <= inputHeight - poolHeight; i += stride[0])
    {
        outputInd[0] = (int)Math.Floor((double)i / stride[0]);
        for (int j = 0; j <= inputWidth - poolWidth; j += stride[1])
        {
            outputInd[1] = (int)Math.Floor((double)j / stride[1]);

            max = float.MinValue;
            for (inputInd[0] = i; inputInd[0] < i + poolHeight;
                inputInd[0]++)
            {
                for (inputInd[1] = j; inputInd[1] < j + poolWidth;
                    inputInd[1]++)
                {
                    float f = inputTensor[inputInd];
                    if (f > max)
                        max = f;
                }
            }
            writeNextLayerInput(outputInd, max);
        }
    }
}

disposeInputTensor();
}

public unsafe class ReLU : Layer
{
    public ReLU(int[] inputTensorDims) : base(inputTensorDims)
    {
        type = "ReLU";
    }

    public override void feedNext()
    {
        outputTensorMemAlloc();
    }
}

```

```

int inputHeight = InputTensorDims[0];
int inputWidth = InputTensorDims[1];
int channelCount = InputTensorDims[2];

float f;
int[] inputInd = new int[] { 0, 0, 0 };
for (inputInd[0] = 0; inputInd[0] < inputHeight; inputInd[0]++)
    for (inputInd[1] = 0; inputInd[1] < inputWidth; inputInd[1]++)
        for (inputInd[2] = 0; inputInd[2] < channelCount;
            inputInd[2]++)
        {
            f = inputTensor[inputInd];
            if (f < 0)
                f = 0;
            writeNextLayerInput(inputInd, f);
        }

disposeInputTensor();
}
}

public unsafe class SoftMax : Layer
{
    public SoftMax(int[] inputTensorDims) : base(inputTensorDims)
    {
        type = "SoftMax";
    }

    public override void feedNext()
    {
        outputTensorMemAlloc();

        float max = float.MinValue;
        for (int i = 0; i < inputTensor.TotalLength; i++)
            if (inputTensor.memPtr[i] > max)
                max = inputTensor.memPtr[i];

        float sum = 0;
        float* nLMR = nextLayer.inputTensor.memPtr;
        for (int i = 0; i < inputTensor.TotalLength; i++)
        {
            nLMR[i] = (float)Math.Exp(inputTensor.memPtr[i] - max);
            sum += nLMR[i];
        }

        for (int i = 0; i < inputTensor.TotalLength; i++)
            nLMR[i] /= sum;
    }
}

```

```
        disposeInputTensor();  
    }  
}
```



EK D: CNNFET – Kaynak Kodlar

```

class KFoldProvider<T1, T2>
{
    public readonly T1[][] data;
    public readonly T2[][] outputs;

    public readonly int k;

    public readonly T1[][][] trainData;
    public readonly T2[][][] trainOutputs;
    public readonly T1[][][] testData;
    public readonly T2[][][] testOutputs;

    public readonly int instanceCount;
    public readonly int attributeCount;
    public readonly int outputCount;

    public readonly int[] shuffledIndexes;

    public KFoldProvider(T1[][] data, T2[][] outputs, int k = 10,
        int randomSeed = int.MinValue)
    {
        instanceCount = data.Length;
        attributeCount = data[0].Length;
        outputCount = outputs[0].Length;

        this.data = new T1[instanceCount][];
        this.outputs = new T2[instanceCount][];

        shuffledIndexes = new int[instanceCount];
        for (int i = 0; i < instanceCount; i++)
        {
            this.data[i] = new T1[attributeCount];
            this.outputs[i] = new T2[outputCount];

            Array.Copy(data[i], this.data[i], attributeCount);
            Array.Copy(outputs[i], this.outputs[i], outputCount);

            shuffledIndexes[i] = i;
        }

        trainData = new T1[k][][];
        trainOutputs = new T2[k][][];
        testData = new T1[k][][];
        testOutputs = new T2[k][][];

        this.k = k;
    }
}

```

```

        shuffle(randomSeed);

        prepareKFolds();
    }

    private void shuffle(int randomSeed = int.MinValue)
    {
        Random randomGenerator;
        if (randomSeed == int.MinValue)
            randomGenerator = new Random(Guid.NewGuid().GetHashCode());
        else
            randomGenerator = new Random(randomSeed);

        float[] randArr = new float[instanceCount];
        for (int i = 0; i < instanceCount; i++)
            randArr[i] = (float)randomGenerator.NextDouble();

        Array.Sort(randArr, shuffledIndexes);
    }

    private void prepareKFolds()
    {
        int vectorCountInEachFold =
            (int)Math.Floor((double)instanceCount / k);

        int vectorCountInKMinus1Fold = (k - 1) * vectorCountInEachFold;
        int vectorCountInTheLastFold =
            instanceCount - vectorCountInKMinus1Fold;

        for (int i = 0; i < k; i++)
        {
            int testInstanceCount;
            if (i == k - 1)
                testInstanceCount = vectorCountInTheLastFold;
            else
                testInstanceCount = vectorCountInEachFold;

            testData[i] = new T1[testInstanceCount][];
            testOutputs[i] = new T2[testInstanceCount][];

            int trainInstanceCount = instanceCount - testInstanceCount;
            trainData[i] = new T1[trainInstanceCount][];
            trainOutputs[i] = new T2[trainInstanceCount][];

            int m = 0;
            for (int j = 0; j < k; j++)
            {
                int lBound = j * vectorCountInEachFold;

```



```

List<int> uniqueLabelList = new List<int>();
for (int i = 0; i < numberOfInstances; i++)
{
    orderedIndexes[i] = i;
    this.labelIndexes[i] = labelIndexes[i];
    this.instances[i] = new float[numberOfFeatures];
    for (int j = 0; j < numberOfFeatures; j++)
        this.instances[i][j] = instances[i][j];
    if (!uniqueLabelList.Contains(labelIndexes[i]))
        uniqueLabelList.Add(labelIndexes[i]);
}
uniqueLabelIndexes = uniqueLabelList.ToArray();
}

public int classify(float[] newInstance, int k)
{
    float[] distances = new float[numberOfInstances];
    Dictionary<int, float> minDistancesPerClass =
        new Dictionary<int, float>();

    int n = uniqueLabelIndexes.Length;
    for (int i = 0; i < n; i++)
        minDistancesPerClass[uniqueLabelIndexes[i]] = float.MaxValue;
    for (int i = 0; i < numberOfInstances; i++)
    {
        float d = calcDistance(instances[i], newInstance);
        distances[i] = d;
        int labelInd = labelIndexes[i];
        if (d < minDistancesPerClass[labelInd])
            minDistancesPerClass[labelInd] = d;
    }

    int[] indexes = (int[])orderedIndexes.Clone();
    Array.Sort(distances, indexes);

    int[] nearestKClasses = new int[k];
    if (k > numberOfInstances) k = numberOfInstances;
    for (int i = 0; i < k; i++)
        nearestKClasses[i] = labelIndexes[indexes[i]];

    Dictionary<int, int> histogram = calcHistogram(nearestKClasses);
    int max = -1;
    int maxInd = 0;
    foreach (KeyValuePair<int, int> kvp in histogram)
    {
        if (kvp.Value == max && minDistancesPerClass[kvp.Key] <
            minDistancesPerClass[maxInd])
        {

```

```

        max = kvp.Value;
        maxInd = kvp.Key;
    }
    else if (kvp.Value > max)
    {
        max = kvp.Value;
        maxInd = kvp.Key;
    }
}
return maxInd;
}

private float calcDistance(float[] instance1, float[] instance2)
{
    float sumOfSquares = 0f;
    for (int i = 0; i < numberOfFeatures; i++)
        sumOfSquares += (float)Math.Pow(instance1[i] - instance2[i], 2);

    return (float)Math.Sqrt(sumOfSquares);
}

private Dictionary<int, int> calcHistogram(int[] array)
{
    int k = array.Length;
    Dictionary<int, int> hist = new Dictionary<int, int>();
    for (int i = 0; i < k; i++)
    {
        int val = array[i];
        if (hist.ContainsKey(val))
            hist[val]++;
        else
            hist.Add(val, 1);
    }

    return hist;
}

}

unsafe class ConvOutputExtractor
{
    private CNN cnn;

    public ConvOutputExtractor(CNN cnn)
    {
        this.cnn = cnn;
    }

    public float[] extract(string imageFile, int layerIndex)
    {

```

```

float[] convOutput = null;
Bitmap b = new Bitmap(imageFile);
cnn.inputLayer.setInput(b, Input.ResizingMethod.ZeroPad);
Layer currentLayer = cnn.inputLayer;
int i = 0;
while (currentLayer.nextLayer != null)
{
    if (i == layerIndex + 1)
    {
        convOutput =
            new float[currentLayer.inputTensor.TotalLength];

        for (int j = 0; j < convOutput.Length; j++)
            convOutput[j] = currentLayer.inputTensor.memPtr[j];
        currentLayer.disposeInputTensor();
        break;
    }

    currentLayer.feedNext();
    currentLayer = currentLayer.nextLayer;
    i += 1;
}

return convOutput;
}
}

class Dataset
{
    public string[] files;
    public string[] classes;
    public int[] classIndexes;
    public string[] distinctClasses;
    public int[] numOfInsPerClass;
    public int distinctClassCount;
    public int count;

    public Dataset(string path, string regexPattern,
        int minNumOfInsPerClass = 0, int maxNumOfInsPerClass = 0)
    {
        FileScanner fs = new FileScanner(new string[]
        { ".jpg", ".jpeg", ".png", ".bmp" });
        string[] unfilteredFiles = fs.scan(path);

        List<string> files = new List<string>();
        List<int> classIndexes = new List<int>();
        List<string> classes = new List<string>();
        List<int> numOfInsPerClass = new List<int>();
    }
}

```

```

List<string> classNames = new List<string>();
foreach (string file in unfilteredFiles)
{
    Regex r = new Regex(regexPattern, RegexOptions.Singleline);
    Match m = r.Match(file);

    string c = m.Groups[1].Value.Replace("\'", "");

    int ind = classNames.IndexOf(c);
    if (ind < 0)
    {
        numOfInsPerClass.Add(0);
        classNames.Add(c);
        ind = classNames.Count - 1;
    }
    else if (maxNumOfInsPerClass > 0 && numOfInsPerClass[ind] >=
maxNumOfInsPerClass)
        continue;

    numOfInsPerClass[ind]++;

    files.Add(file);
    classIndexes.Add(ind);
    classes.Add(c);
}

if (minNumOfInsPerClass > 1)
    for (int i = numOfInsPerClass.Count - 1; i >= 0; i--)
    {
        if (numOfInsPerClass[i] < minNumOfInsPerClass)
        {
            numOfInsPerClass.RemoveAt(i);
            classNames.RemoveAt(i);
            for (int j = files.Count - 1; j >= 0; j--)
                if (classIndexes[j] == i)
                {
                    files.RemoveAt(j);
                    classes.RemoveAt(j);
                    classIndexes.RemoveAt(j);
                }
        }
    }

this.files = files.ToArray();
this.classIndexes = classIndexes.ToArray();
this.classes = classes.ToArray();
this.numOfInsPerClass = numOfInsPerClass.ToArray();
count = files.Count;
distinctClasses = classNames.ToArray();

```

```

distinctClassCount = distinctClasses.Length;

Dictionary<int, int> newClassIndexes = new Dictionary<int, int>();
if(minNumOfInsPerClass>0 && classIndexes.Count > 0)
{
    int k = 0;
    for (int i=0;i< classIndexes.Count; i++)
    {
        int classInd = classIndexes[i];
        int newClassInd;
        if (newClassIndexes.ContainsKey(classInd))
            newClassInd = newClassIndexes[classInd];
        else
        {
            newClassIndexes.Add(classInd, k);
            newClassInd = k;
            k++;
        }
        this.classIndexes[i] = newClassInd;
    }
}

}

class FileScanner
{
    private string[] supportedExtensions;

    public FileScanner(string[] supportedExtensions)
    {
        int l = supportedExtensions.Length;
        this.supportedExtensions = new string[l];
        Array.Copy(supportedExtensions, this.supportedExtensions, l);
    }

    public string[] scan(string path)
    {
        try
        {
            return Directory
                .EnumerateFiles(path, ".*", SearchOption.AllDirectories)
                .Where((file) =>
                {
                    string ext = Path.GetExtension(file).ToLower();
                    return (Array.IndexOf(supportedExtensions, ext) > -1);
                }).ToArray();
        }
        catch (Exception)
        {

```

```

        return new string[0];
    }
}

class FilterVisualizer
{
    private Form frm = null;
    private ComboBox cbFilterIndex;
    private FlowLayoutPanel flp;

    public void visualize(CNN cnn, int layerIndex, int filterIndex)
    {
        Conv layer = (Conv)cnn.layers[layerIndex];

        int[] dims = layer.weights.Dimensions;
        int width = dims[0];
        int height = dims[1];
        int channelCount = dims[2];
        int filterCount = dims[3];

        if (filterIndex >= filterCount)
        {
            MessageBox.Show(null, "Channel index should be a number below "
                + channelCount + ".", "Invalid Channel Index",
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            return;
        }
        if (width < 2 && height < 2)
        {
            MessageBox.Show(null, "Filters in the selected layer cannot be
                visualized since their size is 1x1. (These layers are fully
                connected layers implemented using convolution.)", "Filter
                Size", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            return;
        }

        Bitmap[] bitmaps = new Bitmap[channelCount];
        for (int n = 0; n < channelCount; n++)
        {
            float[,] currFilter = new float[width, height];
            float min = float.MaxValue;
            float max = float.MinValue;
            for (int i = 0; i < width; i++)
                for (int j = 0; j < height; j++)
                {
                    float val = layer.weights[i, j, n, filterIndex];

                    if (val < min)

```

```

        min = val;
        if (val > max)
            max = val;

        currFilter[i, j] = val;
    }

    float max_min = max - min;
    if (max_min == 0)
        max_min = 1;

    Bitmap bmp = new Bitmap(width, height,
        PixelFormat.Format24bppRgb);
    for (int i = 0; i < width; i++)
        for (int j = 0; j < height; j++)
        {
            float val = currFilter[i, j];
            val = 255.0f * (val - min) / max_min;
            int iVal = (int)val;

            bmp.SetPixel(i, j, Color.FromArgb(iVal, iVal, iVal));
        }
    bitmaps[n] = bmp;
}

if (frm == null || frm.IsDisposed)
{
    frm = new Form();
    frm.Size = new Size(430, 300);
    frm.StartPosition = FormStartPosition.CenterScreen;
    frm.ShowIcon = false;
    frm.Text = "Filter Visualizer";

    cbFilterIndex = new ComboBox();
    cbFilterIndex.Dock = DockStyle.Top;
    cbFilterIndex.DropDownStyle = ComboBoxStyle.DropDownList;
    for (int i = 0; i < filterCount; i++)
        cbFilterIndex.Items.Add("Filter index: " + i);
    cbFilterIndex.SelectedIndex = 0;
    cbFilterIndex.SelectedIndexChanged += (object sender,
        EventArgs e) =>
    {
        visualize(cnn, layerIndex, cbFilterIndex.SelectedIndex);
    };

    flp = new FlowLayoutPanel();
    flp.Dock = DockStyle.Fill;
    flp.AutoScroll = true;

```

```

        frm.Controls.Add(flp);
        frm.Controls.Add(cbFilterIndex);

        frm.Show();
    }
    else
        flp.Controls.Clear();

    frm.SuspendLayout();
    for (int n = 0; n < bitmaps.Length; n++)
    {
        PictureBox pb = new PictureBox();
        pb.Width = 50;
        pb.Height = 50;
        pb.BackgroundImageLayout = ImageLayout.Stretch;
        pb.BackgroundImage = bitmaps[n];
        pb.BorderStyle = BorderStyle.FixedSingle;
        flp.Controls.Add(pb);
    }
    frm.ResumeLayout();
}

class Normalizer
{
    public int numberOfInstances;
    public int numberOfFeatures;
    public float[] minValues;
    public float[] maxValues;

    public void normalize(float[][] featureSet)
    {
        numberOfInstances = featureSet.Length;
        numberOfFeatures = featureSet[0].Length;

        minValues = new float[numberOfFeatures];
        maxValues = new float[numberOfFeatures];
        for (int j = 0; j < numberOfFeatures; j++)
        {
            minValues[j] = int.MaxValue;
            maxValues[j] = int.MinValue;
        }

        for (int i = 0; i < numberOfInstances; i++)
            for (int j = 0; j < numberOfFeatures; j++)
            {
                var v = featureSet[i][j];
                if (minValues[j] > v)

```

```

        minValues[j] = v;
        if (maxValues[j] < v)
            maxValues[j] = v;
    }

    normalizeAll(featureSet);
}

public void normalize(float[][] featureSet, List<float> minValues,
List<float> maxValues)
{
    numberOfInstances = featureSet.Length;
    numberOfFeatures = featureSet[0].Length;

    this.minValues = minValues.ToArray();
    this.maxValues = maxValues.ToArray();

    normalizeAll(featureSet);
}

private void normalizeAll(float[][] featureSet)
{
    for (int j = 0; j < numberOfFeatures; j++)
    {
        float max_min = maxValues[j] - minValues[j];
        for (int i = 0; i < numberOfInstances; i++)
            featureSet[i][j] = (featureSet[i][j] - minValues[j]) /
                max_min;
    }
}

static class PearsonCorrelation
{
    private static float correlationBetween(float[] X, float[] Y)
    {
        int n = X.Length;

        float avgX = 0, avgY = 0;
        for (int i = 0; i < n; i++)
        {
            avgX += X[i];
            avgY += Y[i];
        }
        avgX /= n;
        avgY /= n;

        float sum_X_avgX_2 = 0, sum_Y_avgY_2 = 0, sum_X_avgXY_avgY = 0;
        for (int i = 0; i < n; i++)

```

```

{
    float X_avgX = X[i] - avgX;
    float Y_avgY = Y[i] - avgY;

    sum_X_avgX_2 += X_avgX * X_avgX;
    sum_Y_avgY_2 += Y_avgY * Y_avgY;
    sum_X_avgXY_avgY += X_avgX * Y_avgY;
}

float mul = sum_X_avgX_2 * sum_Y_avgY_2;
if (mul == 0)
    return 0;
else
    return (float)(sum_X_avgXY_avgY / Math.Sqrt(Math.Abs(mul)));
}

public static float weightedAvgAbsCorrelation(float[] X, int[] Y,
int distinctClassCount)
{
    int m = distinctClassCount;
    int n = X.Length;
    float[] binarizedClasses = new float[n];
    if (m == 2)
    {
        for (int j = 0; j < n; j++)
            binarizedClasses[j] = Y[j];
        return Math.Abs(correlationBetween(X, binarizedClasses));
    }

    float avgCorr = 0f;
    for (int i = 0; i < m; i++)
    {
        int instanceCount = 0;
        for (int j = 0; j < n; j++)
        {
            if (Y[j] == i)
            {
                binarizedClasses[j] = 1f;
                instanceCount++;
            }
            else
                binarizedClasses[j] = 0f;
        }
        float corr = Math.Abs(correlationBetween(X, binarizedClasses));
        corr *= (float)instanceCount / n;
        avgCorr += corr;
    }
    return avgCorr;
}

```

```

}

class ARFFFileWriter
{
    string filePath;
    string relationName;
    FileStream fileStream;
    public ARFFFileWriter(string filePath, string relationName)
    {
        this.filePath = filePath;
        this.relationName = relationName;
        fileStream = null;
        if (File.Exists(filePath))
            File.Delete(filePath);
    }

    public bool writeAttributeDefinitions(int attributeCount,
    string[] distinctClassNames, List<int> selectedFeatureIndexes)
    {
        try
        {
            fileStream = File.OpenWrite(filePath);
            byte[] byteArray = Encoding.ASCII.GetBytes("@relation " +
            relationName + "\r\n\r\n");
            fileStream.Write(byteArray, 0, byteArray.Length);

            if (selectedFeatureIndexes != null)
                foreach (int i in selectedFeatureIndexes)
                {
                    byteArray = Encoding.ASCII.GetBytes("@attribute attr" +
                    i + " numeric\r\n");
                    fileStream.Write(byteArray, 0, byteArray.Length);
                }
            else
                for (int i = 0; i < attributeCount; i++)
                {
                    byteArray = Encoding.ASCII.GetBytes("@attribute attr" +
                    i + " numeric\r\n");
                    fileStream.Write(byteArray, 0, byteArray.Length);
                }
            byteArray = Encoding.ASCII.GetBytes("@attribute class {'" +
            string.Join("'", distinctClassNames.ToArray()) +
            "'}\r\n\r\n@data\r\n");
            fileStream.Write(byteArray, 0, byteArray.Length);

            return true;
        }
        catch (Exception)
        {

```

```

        closeFile();
        return false;
    }
}

public void writeInstance(float[] data, string instanceClass,
List<int> selectedFeatureIndexes)
{
    StringBuilder sb = new StringBuilder();
    if (selectedFeatureIndexes != null)
        foreach (int i in selectedFeatureIndexes)
            sb.Append(data[i].ToString().Replace(",", ".") + " ");
    else
        for (int i = 0; i < data.Length; i++)
            sb.Append(data[i].ToString().Replace(",", ".") + " ");
    sb.Append("'" + instanceClass + "'\r\n");
    byte[] byteArray = Encoding.ASCII.GetBytes(sb.ToString());
    fileStream.Write(byteArray, 0, byteArray.Length);
}

public void closeFile()
{
    if (fileStream != null)
    {
        fileStream.Close();
        fileStream = null;
    }
}

~ARFFFileWriter()
{
    closeFile();
}
}

class MFileWriter
{
    string filePath;
    string variableName;
    FileStream fileStream;
    public MFileWriter(string filePath, string variableName)
    {
        this.filePath = filePath;
        this.variableName = variableName;
        fileStream = null;
        if (File.Exists(filePath))
            File.Delete(filePath);
    }
}

```

```

public bool writeDefinitions(string[] distinctClassNames)
{
    try
    {
        fileStream = File.OpenWrite(filePath);
        byte[] byteArray = Encoding.ASCII.GetBytes(
            variableName + "_classes = { '" + string.Join("'", ", '",
            distinctClassNames.ToArray()) + "' };\\r\\n" +
            variableName + " = [\\r\\n"
        );
        fileStream.Write(byteArray, 0, byteArray.Length);

        return true;
    }
    catch (Exception)
    {
        closeFile();
        return false;
    }
}

public void writeInstance(float[] data, int instanceClassIndex,
List<int> selectedFeatureIndexes)
{
    StringBuilder sb = new StringBuilder();
    if (selectedFeatureIndexes != null)
        foreach (int i in selectedFeatureIndexes)
            sb.Append(data[i].ToString().Replace(",", ".") + " ");
    else
        for (int i = 0; i < data.Length; i++)
            sb.Append(data[i].ToString().Replace(",", ".") + " ");

    sb.Append(instanceClassIndex.ToString() + "\\r\\n");
    byte[] byteArray = Encoding.ASCII.GetBytes(sb.ToString());
    fileStream.Write(byteArray, 0, byteArray.Length);
}

public void closeFile()
{
    byte[] byteArray = Encoding.ASCII.GetBytes("];\\r\\n%Each row
represents one instance and ends with class index (starting from
zero)\\r\\n\\r\\n");
    fileStream.Write(byteArray, 0, byteArray.Length);

    if (fileStream != null)
    {
        fileStream.Close();
        fileStream = null;
    }
}

```

```

    }

    ~MFileWriter()
    {
        closeFile();
    }
}

public partial class frmMain : Form
{
    const int FORM_PADDING = 11;

    DateTime dateTime;
    CNN cnn;
    int lastConvLayerInd = -1;
    ConvOutputExtractor convOutputExtractor;
    Dataset dataset;
    float[][] featureSet;
    Random random;

    public unsafe frmMain()
    {
        random = new Random(DateTime.Now.Millisecond);
        InitializeComponent();
        foreach (Control c in tabControl.TabPages)
        {
            if (c != tpMain && c != tpHelp)
                c.Enabled = false;
        }

        cbKVal.SelectedIndex = 0;
        cbClassifier.SelectedIndex = 0;

        pnlProgress.Top = tabControl.Top;
        pnlProgress.Left = tabControl.Left;
        pnlProgress.Width = tabControl.Width;
        pnlProgress.Height = tabControl.Height;

        lvCorrelations.Left = ClientSize.Width - lvCorrelations.Width -
            FORM_PADDING;
        lvCorrelations.Top = FORM_PADDING;
        lvCorrelations.Height = ClientSize.Height - FORM_PADDING * 2;

        makeDoubleBuffered(pnlProgress);

        Version v = Assembly.GetExecutingAssembly().GetName().Version;
        Text += " v" + v.Major + "." + v.Minor;
    }
}

```

```

public void makeDoubleBuffered(Control ctrl)
{
    ctrl.GetType()
        .GetProperty("DoubleBuffered", BindingFlags.Instance |
            BindingFlags.NonPublic)
        .SetValue(ctrl, true, null);
}

public string round(float f, int decimals)
{
    return Math.Round(f, decimals).ToString("0.#####")
        .Replace(',', '.');
}

void enableTabControl()
{
    if (InvokeRequired)
    {
        Invoke(new Action(() => enableTabControl()));
        return;
    }
    tabControl.Enabled = true;
}

private string leftPad(string str, int len)
{
    var i = str.Length;
    if (i >= len)
        return str;
    else
        return new string(' ', len - i) + str;
}

private DateTime tic()
{
    dateTime = DateTime.Now;
    return dateTime;
}

private double toc()
{
    return Math.Round((DateTime.Now - dateTime).TotalSeconds, 3);
}

private void appendText(string text)
{
    if (InvokeRequired)
    {
        Invoke(new Action(() => appendText(text)));
        return;
    }
}

```

```

    }

    txtLog.AppendText(text);
}

private void appendLine(string text, string prefix = "--> ")
{
    if (InvokeRequired)
    {
        Invoke(new Action(() => appendLine(text, prefix)));
        return;
    }

    txtLog.AppendText("\r\n" + prefix + text);
}

bool breakLoops = false;
void showProgressPanel(bool noProgress = false)
{
    if (InvokeRequired)
    {
        Invoke(new Action(() => showProgressPanel()));
        return;
    }
    tabControl.Enabled = false;

    lblTerminate.Text = "Stop";
    breakLoops = false;
    pbProgress.Value = 0;
    pbProgress.Style = noProgress ? ProgressBarStyle.Marquee :
    ProgressBarStyle.Continuous;
    lblProgress.Text = noProgress ? "Please wait..." : "... 0%";
    pnlProgress.Visible = true;
}

float prevPercent, prevPercent2;
private void progress(int i, int n)
{
    if (InvokeRequired)
    {
        Invoke(new Action(() => progress(i, n)));
        return;
    }

    if (i == 0)
    {
        prevPercent = -1;
        prevPercent2 = -1;
    }
}

```

```

int j = i + 1;
float percent = j * 100f / n;
float diff = percent - prevPercent;
float diff2 = percent - prevPercent2;

if (diff > 0.1)
{
    string str =
        round(percent, 3) + "%" + "\r\n\r\n" + j + " / " + n;
    pbProgress.Value = (int)percent;
    lblProgress.Text = str;
    prevPercent = percent;
    Application.DoEvents();
}
}

private void finishedIn()
{
    if (breakLoops)
        appendLine("Interrupted by the user after " +
            round((float)toc(), 3) + " seconds");
    else
        appendLine("Finished in " + round((float)toc(), 3) +
            " seconds");
    Invoke(new Action(() => pnlProgress.Visible = false));
}

private void Form1_Load(object sender, EventArgs e)
{
    CBLAS.detectCBLAS();
    txtLog.Text = "--> Checking BLAS support...";
    Application.DoEvents();
    appendLine("OpenBLAS: " + (CBLAS.openbAvailable ? "" : "not ") +
        "available", "");
    appendLine("Intel MKL: " + (CBLAS.imklAvailable ? "" : "not ") +
        "available", "");
    if (!CBLAS.openbAvailable && !CBLAS.imklAvailable)
        appendLine("The application is written to be able to run
            standalone. But you can improve the performance drastically,
            using a BLAS library. Please visit GIT repository of this
            application for more info.", "");
    appendLine("Ready");
    cbRegex.SelectedIndex = 0;
}

private void txtLog_DoubleClick(object sender, EventArgs e)
{
    if (ModifierKeys.HasFlag(Keys.Control))

```

```

        txtLog.Clear();
    }

    private void btnBrowseCeNiNFile_Click(object sender, EventArgs e)
    {
        OpenFileDialog opf = new OpenFileDialog();
        opf.Filter = "CeNiN file|*.cenin";
        if (opf.ShowDialog() != DialogResult.OK) return;

        txtCeNiNFile.Text = opf.FileName;

        appendLine("Loading CNN file...");

        tic();
        tabControl.Enabled = false;

        new Thread(() =>
        {
            cnn = new CNN(opf.FileName);
            appendLine("Total number of weights: " +
            cnn.totalWeightCount, "");
            appendLine("Total number of biases: " + cnn.totalBiasCount, "");
            appendLine(cnn.layerCount + "+2 layers were loaded in " +
            toc() + " seconds");

            Invoke(new Action(() => cbLayerList.Items.Clear()));
            int i = 1;
            lastConvLayerInd = -1;
            foreach (Layer layer in cnn.layers)
            {
                string str = layer.type;
                if (!(layer is Input || layer is Output))
                {
                    str = "#" + i + " " + str;
                    int[] dims = (int[])layer.InputTensorDims.Clone();
                    if (layer.pad != null)
                    {
                        {
                            dims[0] -= layer.pad[0] + layer.pad[1];
                            dims[1] -= layer.pad[2] + layer.pad[3];
                        }
                        str += " (input:[" + string.Join(", ", dims) + "]);"
                        i++;
                    }
                }
                if (layer is Conv)
                {
                    int[] dims = ((Conv)layer).weights.Dimensions;
                    str += " filters:[" + string.Join(", ", dims) + "];"
                    lastConvLayerInd = i - 1;
                }
            }
        })
    }

```

```

        str += " )";
        Invoke(new Action(() => cbLayerList.Items.Add(str)));
    }
    if (lastConvLayerInd != -1)
        Invoke(new Action(() => cbLayerList.SelectedIndex =
            lastConvLayerInd));

    convOutputExtractor = new ConvOutputExtractor(cnn);

    finishedIn();
    enableTabControl();

}).Start();
}

private void scanFiles()
{
    string regexPattern = cbRegex.Text;
    if (cbRegex.SelectedIndex >= 0)
        regexPattern = regexPattern.Substring(0,
            regexPattern.IndexOf("$") + 1);
    try
    {
        Regex r = new Regex(regexPattern, RegexOptions.Singleline);
    }
    catch (Exception)
    {
        MessageBox.Show(this, "The regex value written for the class
            name pattern is invalid!", "Error", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        return;
    }

    string selectedPath = txtImagesPath.Text;

    appendLine("Scanning image files...");

    tic();
    tabControl.Enabled = false;
    int maxNumOfInsPerClass = cbNofInstanceLimit.Checked ?
        (int)nudNofInstanceLimit.Value : 0;
    int minNumOfInsPerClass = cbMinSamplePerClass.Checked ?
        (int)nudMinSamplePerClass.Value : 0;

    new Thread(() =>
    {
        dataset = new Dataset(selectedPath, regexPattern,
            minNumOfInsPerClass, maxNumOfInsPerClass);
    }

```

```

        if (minNumOfInsPerClass > 0)
            appendLine("Minimum number of instances per class: " +
                minNumOfInsPerClass, "");
        if (maxNumOfInsPerClass > 0)
            appendLine("Maximum number of instances per class: " +
                maxNumOfInsPerClass, "");
        appendLine("Number of classes: " +
            dataset.distinctClasses.Length, "");
        appendLine("Number of images: " + dataset.count, "");

        int maxStrLen = 0;
        for (int i = 0; i < dataset.distinctClassCount; i++)
        {
            int len = dataset.distinctClasses[i].Length;
            if (maxStrLen < len)
                maxStrLen = len;
        }

        for (int i = 0; i < dataset.distinctClassCount; i++)
            appendLine(" " + leftPad(dataset.distinctClasses[i],
                maxStrLen) + ": " + dataset.numOfInsPerClass[i], "");

        if (dataset.count < 2 || dataset.distinctClassCount < 2)
        {
            appendLine("Both the number of images and the number of
                classes found in the directory should be greater than 1.");
            dataset = null;
        }

        finishedIn();
        enableTabControl();

    }).Start();
}

private void btnBrowseDatasetDirectory_Click(object sender, EventArgs e)
{
    FolderBrowserDialog fbd = new FolderBrowserDialog();
    if (fbd.ShowDialog() != DialogResult.OK) return;

    txtImagesPath.Text = fbd.SelectedPath;

    scanFiles();
}

bool isConvSelected()
{
    string type = cbLayerList.Text;
    if (type.Contains("#"))

```

```

        type = type.Substring(type.IndexOf(' ') + 1).Trim();
    bool b = type.StartsWith("Conv");
    if (!b)
        MessageBox.Show(this, "Please select a convolution layer",
            "Invalid Layer Type", MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);
    return b;
}

private void btnVisualize_Click(object sender, EventArgs e)
{
    if (isConvSelected())
        (new FilterVisualizer()).visualize(cnn,
            cbLayerList.SelectedIndex, 0);
}

private void btnExtract_Click(object sender, EventArgs e)
{
    if (!isConvSelected())
        return;
    if (dataset is null)
    {
        MessageBox.Show(this, "Please select a dataset directory.",
            "Empty Dataset", MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);
        return;
    }

    int layerIndex = cbLayerList.SelectedIndex;

    selectedFeatureIndexes = null;
    lvCorrelations.Items.Clear();

    appendLine("Extracting...");
    showProgressPanel();

    tic();

    int n = dataset.files.Length;
    featureSet = new float[n][];

    new Thread(() =>
    {
        ParallelOptions po = new ParallelOptions();
        po.MaxDegreeOfParallelism = Environment.ProcessorCount;
        Mutex m = new Mutex();
        int j = 0;
        Parallel.For(0, dataset.count, po, (int i, ParallelLoopState
            state) =>

```

```

    {
        featureSet[i] =
            convOutputExtractor.extract(dataset.files[i], layerIndex);
        m.WaitOne();
        progress(j, n);
        j++;
        m.ReleaseMutex();
        if (breakLoops)
        {
            featureSet = null;
            state.Break();
        }
    });
    if (!breakLoops && featureSet != null && featureSet.Length > 0)
        appendLine("Number of features extracted per each
            instance: " + featureSet[0].Length, "");
    finishedIn();
    enableTabControl();
}).Start();
}

private bool checkFeatureSet()
{
    if (featureSet is null)
    {
        MessageBox.Show(this, "There is no extracted feature set yet.",
            "Warning", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return false;
    }
    return true;
}

private void export(bool arff)
{
    if (!checkFeatureSet())
        return;

    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = arff ? "ARFF file|*.arff" : "Matlab script|*.m";
    if (sfd.ShowDialog() != DialogResult.OK) return;

    appendLine("Exporting feature vectors...");
    showProgressPanel();

    tic();
    if (arff)
    {
        ARFFFileWriter arffCreator = new ARFFFileWriter(sfd.FileName,
            "CNNFET" + random.Next(10000, 99999));
    }
}

```

```

arffCreator.writeAttributeDefinitions(featureSet[0].Length,
dataset.distinctClasses, selectedFeatureIndexes);

int n = featureSet.Length;
for (int i = 0; i < n; i++)
{
    arffCreator.writeInstance(featureSet[i], dataset.classes[i],
selectedFeatureIndexes);
    progress(i, n);
    if (breakLoops)
        break;
}
arffCreator.closeFile();
}
else
{
    MFileWriter mFileCreator = new MFileWriter(sfd.FileName,
"cnnfet" + random.Next(10000, 99999));
mFileCreator.writeDefinitions(dataset.distinctClasses);

int n = featureSet.Length;
for (int i = 0; i < n; i++)
{
    mFileCreator.writeInstance(featureSet[i],
dataset.classIndexes[i], selectedFeatureIndexes);
    progress(i, n);
    if (breakLoops)
        break;
}
mFileCreator.closeFile();
}

finishedIn();
enableTabControl();
}

private void btnBrowse3_Click(object sender, EventArgs e)
{
    if (cnn is null)
    {
        MessageBox.Show(this, "Please load a CNN file first.",
"Warning", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }

    OpenFileDialog opf = new OpenFileDialog();
    opf.Filter = "Image files|*.bmp;*.jpeg;*.jpg;*.png";
    if (opf.ShowDialog() != DialogResult.OK) return;
    Bitmap b = new Bitmap(opf.FileName);

```

```

cnn.inputLayer.setInput(b, Input.ResizingMethod.ZeroPad);
tic();
Layer currentLayer = cnn.inputLayer;
int i = 0;
while (currentLayer.nextLayer != null)
{
    if (i == 0)
        appendLine("Passing the image through the layers...");
    else
        appendLine("Layer " + i + " (" + currentLayer.type
            + ") input:[" +
            string.Join(",", currentLayer.InputTensorDims) + "]" +
            (currentLayer is Conv ? " width,height,channels,filters:[" +
            string.Join(",", ((Conv)currentLayer).weights.Dimensions) +
            "]" + " biases:[" +
            string.Join(",", ((Conv)currentLayer).biases.Dimensions) +
            "]" : "");

        Application.DoEvents();

        currentLayer.feedNext();
        currentLayer = currentLayer.nextLayer;
        i += 1;
}

Output outputLayer = (Output)currentLayer;
finishedIn();

string decision = outputLayer.getDecision();
string hLine = new string('-', 50);
appendLine(hLine, "");
appendLine("THE HIGHEST 3 PROBABILITIES: ", "");
appendLine(hLine, "");
for (i = 0; i < 3; i++)
    appendLine(" #" + (i + 1) + " " + outputLayer.sortedClasses[i]
        + " (" + round(outputLayer.probabilities[i], 3) + ")", "");
appendLine(hLine, "");
appendLine("DECISION: " + decision);
appendLine(hLine, "");
}

private void btnClassify_Click(object sender, EventArgs e)
{
    int foldCount = (int)nudFolds.Value;
    int kNNk = int.Parse(cbKVal.SelectedItem.ToString());
    appendLine("Performing classification (" + foldCount +
        "-fold cross-validation) using "
        + (selectedFeatureIndexes != null ? selectedFeatureIndexes.Count
            : featureSet[0].Length) + " features on "

```

```

        + featureSet.Length + " instances...\r\n");
showProgressPanel();
tic();
ParallelOptions po = new ParallelOptions();
po.MaxDegreeOfParallelism = Environment.ProcessorCount;
new Thread(() =>
{
    int[][] outputs = new int[dataset.count][];
    for (int i = 0; i < dataset.count; i++)
        outputs[i] = new int[1] { dataset.classIndexes[i] };

    KFoldProvider<float, int> kfp;
    if (selectedFeatureIndexes != null)
    {
        appendLine("Preparing feature subset. This may take a while...", "");
        int numberOfInstances = featureSet.Length;
        int numberOfFeatures = featureSet[0].Length;
        float[][] featureSubset = new float[numberOfInstances][];
        for (int i = 0; i < numberOfInstances; i++)
            featureSubset[i] =
                new float[selectedFeatureIndexes.Count];
        int k = 0;
        for (int j = 0; j < numberOfFeatures; j++)
        {
            if (!selectedFeatureIndexes.Contains(j))
                continue;
            for (int i = 0; i < numberOfInstances; i++)
                featureSubset[i][k] = featureSet[i][j];
            k++;
        }
        appendLine("The subset is ready", "");

        kfp = new KFoldProvider<float, int>(featureSubset, outputs,
            foldCount, cbRandRandSeed.Checked ? int.MinValue : 1);
    }
    else
        kfp = new KFoldProvider<float, int>(featureSet, outputs,
            foldCount, cbRandRandSeed.Checked ? int.MinValue : 1);

    int[,] confusionMatrix = new int[dataset.distinctClassCount,
        dataset.distinctClassCount];
    Array.Clear(confusionMatrix, 0, dataset.distinctClassCount *
        dataset.distinctClassCount);

    int total = 0;
    int tp_tn = 0;
    for (int j = 0; j < kfp.k; j++)
    {

```

```

float[][] trainData = kfp.trainData[j];
int[][] trainOutputs = kfp.trainOutputs[j];

float[][] testData = kfp.testData[j];
int[][] testOutputs = kfp.testOutputs[j];

appendLine("Fold " + (j + 1) + ": " + trainData.Length +
" training, " + testData.Length + " test instances", "");

int[] labels = trainOutputs.Select<int[], int>(innerArr =>
innerArr[0]).ToArray<int>();

KNNClassifier knnc = new KNNClassifier(trainData, labels);

Mutex m = new Mutex();
Parallel.For(0, testData.Length, po, (int i,
ParallelLoopState state) =>
{
    int predictedClassInd =
    knnc.classify(testData[i], knnc);
    int realClassInd = testOutputs[i][0];

    confusionMatrix[realClassInd, predictedClassInd]++;

    m.WaitOne();
    if (predictedClassInd == realClassInd)
        tp_tn++;
    progress(total, dataset.count);
    total++;
    m.ReleaseMutex();

    if (breakLoops)
        state.Break();
});
if (breakLoops)
    break;
}

if (!breakLoops)
{
    if (dataset.distinctClassCount < 27)
    {
        string chars = "";
        for (char c = 'a'; c <= 'z'; c++)
            chars += c;

        int maxNumberStrLen = dataset.count.ToString().Length;

        string line = "\r\nCONFUSION MATRIX\r\n";

```

```

        for (int i = 0; i < dataset.distinctClassCount; i++)
            line += leftPad(chars.Substring(i, 1),
                maxNumberStrLen) + " ";
        line += "<-- classified as\r\n";
        for (int i = 0; i < dataset.distinctClassCount; i++)
        {
            for (int j = 0; j < dataset.distinctClassCount; j++)
            {
                line += leftPad(confusionMatrix[i,
                    j].ToString(), maxNumberStrLen) + " ";
            }
            line += "| " + chars.Substring(i, 1) + " = " +
                dataset.distinctClasses[i] + "\r\n";
        }
        appendLine(line, "");
    }

    appendLine("Accuracy: " +
        round(100f * (float)tp_tn / total, 6) + "%" +
        " [" + tp_tn + " / " + total + "]\r\n", "");
}
finishedIn();
enableTabControl();
}).Start();
}

private void tabControl_SelectedIndexChanged(object sender, EventArgs e)
{
    Control selectedTab = tabControl.SelectedTab;
    if (selectedTab == tpFeatureSelection)
    {
        lvCorrelations.Visible = true;
        tabControl.Width = ClientSize.Width - lvCorrelations.Width - 3 *
            FORM_PADDING;
        txtLog.Width = ClientSize.Width - lvCorrelations.Width - 3 *
            FORM_PADDING;
        pnlProgress.Width = tabControl.Width;
        pnlProgress.Height = tabControl.Height;

        if (!checkFeatureSet())
        {
            selectedTab.Enabled = false;
            return;
        }
        else
            selectedTab.Enabled = true;

        if (lvCorrelations.Items.Count < 1)

```

```

{
    tic();
    appendLine("Calculating correlations...");
    showProgressPanel();

    int instanceCount = featureSet.Length;
    int attributeCount = featureSet[0].Length;

    float[] correlations = new float[attributeCount];

    float[] column = new float[instanceCount];
    for (int j = 0; j < attributeCount; j++)
    {
        for (int i = 0; i < instanceCount; i++)
            column[i] = featureSet[i][j];
        correlations[j] =
            PearsonCorrelation.weightedAvgAbsCorrelation(column,
                dataset.classIndexes, dataset.distinctClassCount);
        progress(j, attributeCount);
        if (breakLoops)
            break;
    }
    if (breakLoops)
    {
        selectedFeatureIndexes = null;
        lvCorrelations.Items.Clear();
        tpFeatureSelection.Enabled = false;
    }
    else
    {
        prepareInterface(correlations);
        if (attributeCount > 500)
            appendLine("Since there are so many features, only
                the first 500 features are shown.", "");
    }
    finishedIn();
    enableTabControl();
}
}
else
{
    lvCorrelations.Visible = false;
    tabControl.Width = ClientSize.Width - FORM_PADDING * 2;
    txtLog.Width = ClientSize.Width - FORM_PADDING * 2;
    pnlProgress.Width = tabControl.Width;
    pnlProgress.Height = tabControl.Height;

    if (selectedTab == tpExport ||
        selectedTab == tpClassification ||

```

```

        selectedTab == tpNormalization)
    {
        if (!checkFeatureSet())
        {
            selectedTab.Enabled = false;
            return;
        }
        else
            selectedTab.Enabled = true;

        if (selectedTab == tpClassification)
        {
            int max = dataset.count;
            nudFolds.Maximum = max;
            nudFolds.Value = (dataset.count >= 10 ? 10 : max);
        }
    }
}

private void btnExport_Click(object sender, EventArgs e)
{
    export(rbWeka.Checked);
}

private void lblTerminate_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    if (breakLoops)
        return;
    if (MessageBox.Show(this, "The operations will be terminated before
being completed. Do you want to interrupt?", "Please Confirm",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) !=
DialogResult.Yes)
        return;
    breakLoops = true;
    lblTerminate.Text = "Interrupting...";
}

private void cbNofInstanceLimit_CheckedChanged(object sender,
EventArgs e)
{
    nudNofInstanceLimit.Enabled = cbNofInstanceLimit.Checked;
}

private void cbMinSamplePerClass_CheckedChanged(object sender,
EventArgs e)
{
    nudMinSamplePerClass.Enabled = cbMinSamplePerClass.Checked;
}

```

```

}

private void frmMain_FormClosing(object sender,
FormClosingEventArgs e)
{
    Environment.Exit(0);
}

private void rtbHelp_LinkClicked(object sender, LinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("explorer.exe", e.LinkText);
}

#region FeatureSelection

private int[] sortedIndexes;
private int[] sortedIndexes_invMapping;
private float[] sortedAvgAbsCorrelations;
private List<int> selectedFeatureIndexes;
private int featureCount;

void prepareInterface(float[] avgAbsCorrelations)
{
    featureCount = avgAbsCorrelations.Length;

    sortedAvgAbsCorrelations = new float[featureCount];
    sortedIndexes = new int[featureCount];
    for (int i = 0; i < featureCount; i++)
    {
        sortedAvgAbsCorrelations[i] = avgAbsCorrelations[i];
        sortedIndexes[i] = i;
    }
    Array.Sort(sortedAvgAbsCorrelations, sortedIndexes);
    Array.Reverse(sortedIndexes);
    Array.Reverse(sortedAvgAbsCorrelations);

    sortedIndexes_invMapping = new int[featureCount];

    int featuresToShow = featureCount;
    if (featuresToShow > 500)
        featuresToShow = 500;
    lvCorrelations.BeginUpdate();
    for (int i = 0; i < featureCount; i++)
    {
        if (i < featuresToShow)
        {
            int j = sortedIndexes[i];
            ListViewItem lvi = lvCorrelations.Items.Add("Feature" + j);
            lvi.SubItems.Add(round(avgAbsCorrelations[j], 6));
        }
    }
}

```

```

    }

    sortedIndexes_invMapping[sortedIndexes[i]] = i;
}
lvCorrelations.Columns[0].Width = -2;
lvCorrelations.Columns[1].Width = -2;
lvCorrelations.EndUpdate();

nudFeatureCount.Maximum = featureCount;
nudFeatureCount.Value = featureCount;
nudThreshold.Value = 0.35m;
rbThreshold.Checked = true;
}

private bool disableCheckBoxChanges = true;
private void btnApplyFS_Click(object sender, EventArgs e)
{
    selectedFeatureIndexes = new List<int>();

    int fc = (int)nudFeatureCount.Value;
    float threshold = (float)nudThreshold.Value;
    List<string> featureList = new List<string>();
    if (rbFeatures.Checked)
    {
        string[] featuresToSearch = rtbFeatures.Text.Split(new char[] {
            ',' }, StringSplitOptions.RemoveEmptyEntries).ToArray();
        foreach (string f in featuresToSearch)
        {
            string trimmed = f.Trim();
            if (!featureList.Contains(trimmed))
                featureList.Add(trimmed);
        }
    }
    disableCheckBoxChanges = false;
    lvCorrelations.BeginUpdate();
    for (int i = 0; i < featureCount; i++)
    {
        bool check;
        if (rbFeatures.Checked)
        {
            check = featureList.Contains(i.ToString());
            if (check)
                selectedFeatureIndexes.Add(i);
            int j = sortedIndexes_invMapping[i];
            if (j < 500)
                lvCorrelations.Items[j].Checked = check;
        }
        else
        {

```

```

        if (rbFeatureCount.Checked)
            check = i < fc;
        else// if (rbThreshold.Checked)
            check = sortedAvgAbsCorrelations[i] >= threshold;
        if (check)
            selectedFeatureIndexes.Add(sortedIndexes[i]);
        if (i < 500)
            lvCorrelations.Items[i].Checked = check;
    }
}
lvCorrelations.EndUpdate();
disableCheckBoxChanges = true;
selectedFeatureIndexes.Sort();
rtbFeatures.Text = string.Join(", ", selectedFeatureIndexes);

if (selectedFeatureIndexes.Count < 1)
{
    appendLine("No feature is selected. All the features (" +
        featureCount + ") will be included.");
    selectedFeatureIndexes = null;
}
else if (selectedFeatureIndexes.Count == featureCount)
{
    appendLine("The number of selected features is equal to the
        total number of features. All the features (" + featureCount +
        ") will be included.");
    selectedFeatureIndexes = null;
}
else
    appendLine(selectedFeatureIndexes.Count + " features are
        selected. The others will be excluded.");
}

private void listView1_ItemCheck(object sender, ItemCheckEventArgs e)
{
    if (disableCheckBoxChanges)
        e.NewValue = e.CurrentValue;
}

private void btnRescan_Click(object sender, EventArgs e)
{
    if (txtImagesPath.TextLength > 0)
        scanFiles();
    else
        MessageBox.Show(this, "Please select a dataset directory
            first.", "Dataset", MessageBoxButtons.OK,
            MessageBoxIcon.Warning);
}

```

```

private void btnApplyNorm_Click(object sender, EventArgs e)
{
    tic();
    appendLine("Applying min-max normalization...");
    showProgressPanel(true);

    string rtbNormMinValsText = rtbNormMinVals.Text;
    string rtbNormMaxValsText = rtbNormMaxVals.Text;
    new Thread(() =>
    {
        char decimalSeparator = CultureInfo.CurrentCulture.NumberFormat
            .NumberDecimalSeparator[0];
        Normalizer normalizer = new Normalizer();
        if (rbNormRecalc.Checked)
        {
            normalizer.normalize(featureSet);
            Invoke(new Action(() =>
            {
                rtbNormMinVals.Text = string.
                    Join("| ", normalizer.minValues).
                    Replace(decimalSeparator, '.').Replace('|', ',');
                rtbNormMaxVals.Text =
                    string.Join("| ", normalizer.maxValues).
                    Replace(decimalSeparator, '.').Replace('|', ',');
            }));
        }
        else //if (rbNormCalc.Checked)
        {
            List<float> minValues = new List<float>();
            string[] strArr = rtbNormMinValsText.Split(new char[] { ',', ' ' },
                StringSplitOptions.RemoveEmptyEntries).ToArray();
            foreach (string str in strArr)
            {
                string trimmed = str.Replace('.', decimalSeparator).Trim();
                float f;
                if (float.TryParse(trimmed, out f))
                    minValues.Add(f);
            }
            List<float> maxValues = new List<float>();
            strArr = rtbNormMaxValsText.Split(new char[] { ',', ' ' },
                StringSplitOptions.RemoveEmptyEntries).ToArray();
            foreach (string str in strArr)
            {
                string trimmed = str.Replace('.', decimalSeparator).Trim();
                float f;
                if (float.TryParse(trimmed, out f))
                    maxValues.Add(f);
            }
        }
    })
}

```

```

    }
    if (minValues.Count != maxValues.Count || minValues.Count !=
        featureSet[0].Length)
        appendLine("The length of the normalization vectors (min
            and max) is not equal to the number of features!");
    else
        normalizer.normalize(featureSet, minValues, maxValues);
    }
    finishedIn();
    enableTabControl();
}).Start();
}

private void rtbFeatures_DoubleClick(object sender, EventArgs e)
{
    int len = rtbFeatures.Text.Length;
    if (len < 1)
        return;
    rtbFeatures.Select(0, len);
    Clipboard.SetText(rtbFeatures.Text);
    MessageBox.Show(this, "Copied to the clipboard", "Copied",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}

#endregion
}

```

KAYNAKLAR

- Al-Yasriy H. F., Al-Husieny M. S., Mohsen F. Y., Khalil E. A. ve Hassan Z. S. Diagnosis of Lung Cancer Based on CT Scans Using CNN, *IOP Conference Series: Materials Science and Engineering*, 928, 2020.
- Bay, H., Tuytelaars, T. ve Van Gool, L. (2006, May). Surf: Speeded up robust features. *European conference on computer vision*, 404-417, Springer, Berlin, Heidelberg.
- Bobadilla, J., Fernando, O., ve Antonio H.. A collaborative filtering similarity measure based on singularities. *Information Processing & Management* 48.2 (2012): 204-217.
- Busato, A., Servi, M., Volpe, Y., Furferi, R. ve Scorianz, M. (2023). Precise: A web-based 3D visualization and manipulation application for surgical planning of tumour resection. *SoftwareX*, 22, 101398.
- Chang C., C., ve Lin C., J. (2011). LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 27, 1-27.
- Chen, G., Sun, P., & Shang, Y. (2017, November). Automatic fish classification system using deep learning. *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 24-29, IEEE.
- Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B. ve Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.
- Corradini, F., Beriot, N., Huerta-Lwanga, E. ve Geissen, V. (2021). uFTIR: An R package to process hyperspectral images of environmental samples captured with μ FTIR microscopes. *SoftwareX*, 16, 100857.
- Das, D. K., Chakraborty, C., Mitra, B., Maiti, A. K. ve Ray, A. K. (2013). Quantitative microscopy approach for shape-based erythrocytes characterization in anaemia. *Journal of microscopy*, 249(2), 136-149.
- De Baets, L., Develder, C., Dhaene, T. ve Deschrijver, D. (2017, October). Automated classification of appliances using elliptical fourier descriptors. *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 153-158, IEEE.

- Frank, E., Hall, M., Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.
- González-Lozoya, S. M., de la Calleja, J., Pellegrin, L., Escalante, H. J., Medina, M. A. ve Benitez-Ruiz, A. (2020). Recognition of facial expressions based on CNN features. *Multimedia tools and applications*, 79, 13987-14007.
- González-Ruiz, V. ve Fernández, J. J. (2023). FlowDenoising: Structure-preserving denoising in 3D electron microscopy (3DEM). *SoftwareX*, 23, 101413.
- Gyori, B. M., Venkatachalam, G., Thiagarajan, P. S., Hsu, D., & Clement, M. V. (2014). OpenComet: an automated tool for comet assay image analysis. *Redox Biology*, 2, 457-465.
- Hosseini, A., Eshraghi, M. A., Taami, T., Sadeghsalehi, H., Hoseinzadeh, Z., Ghaderzadeh, M. ve Rafiee, M. (2023). A mobile application based on efficient lightweight CNN model for classification of B-ALL cancer from non-cancerous cells: a design and implementation study. *Informatics in Medicine Unlocked*, 39, 101244.
- Hubel, D. H. ve Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1), 106-154.
- İnternet: Srinivasan S. (2020). Fish Species Image Data, URL: <https://www.kaggle.com/datasets/sripaadsrinivasan/fish-species-image-data>, Son Erişim Tarihi: 24.05.2023.
- İnternet: Weisstein E. (2004), Fourier Series, URL: <https://mathworld.wolfram.com/FourierSeries.html>, Son Erişim Tarihi: 20.09.2020.
- Jung, S. K. (2021). AniLength: GUI-based automatic worm length measurement software using image processing and deep neural network. *SoftwareX*, 15, 100795.
- Khan, S. H., Abbas, Z. ve Rizvi, S. D. (2019, February). Classification of diabetic retinopathy images based on customised CNN architecture. *2019 Amity international conference on artificial intelligence (AICAI)*, 244-248, IEEE.

- Kim, H. K., Kim, J. D., Sim, D. G. ve Oh, D. I. (2000, July). A modified Zernike moment shape descriptor invariant to translation, rotation and scale for similarity-based image retrieval. *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia* 00TH8532(1), 307-310, IEEE.
- Kodors, S., Lacis, G., Sokolova, O., Zhukovs, V., Apeinans, I. ve Bartulsons, T. (2021). Apple Scab Detection using CNN and Transfer Learning. *Agronomy Research*, 19(2), 507–519.
- Końca, K., Lankoff, A., Banasik, A., Lisowska, H., Kuszewski, T., Gózdź, S., Wojcik, A. (2003). A cross-platform public domain PC image-analysis program for the comet assay. *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, 534(1-2), 15-20.
- Kutlu, Y. ve Turan, C. (2018). An Intelligent Software for Measurements of Biological Materials: BioMorph. *Natural and Engineering Sciences*, 3(2), 225-233.
- López-Sampson, A. ve Page, T. (2018). Elliptical Fourier descriptors of leaf outlines: a tool to discriminate among Aquilaria species (Thymelaeaceae). *Silvae Genetica*, 67(1), 89-92.
- Mathur, M. ve Goel, N. (2021). FishResNet: Automatic fish classification approach in underwater scenario. *SN Computer Science*, 2(4), 273.
- Mei, Y. ve Androutsos, D. (2009). Robust affine invariant region-based shape descriptors: the ICA Zernike moment shape descriptor and the whitening Zernike moment shape descriptor. *IEEE Signal Processing Letters*, 16(10), 877-880.
- Nawaz, H., Maqsood, M., Afzal, S., Aadil, F., Mehmood, I. ve Rho, S. (2021). A deep feature-based real-time system for Alzheimer disease stage detection. *Multimedia Tools and Applications*, 80, 35789-35807.
- Nilsback, M. E. ve Zisserman, A. (2018). Automated flower classification over a large number of classes. *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 722-729, IEEE.

- Nouaze, J. C. ve Sikati, J. (2023). YOLO-AppleScab: A Deep Learning Approach for Efficient and 2 Accurate Apple Scab Detection in Varied Lighting Conditions 3 Using CARAFE-enhanced YOLOv7. *Biology and Life Sciences Forum 2023*, 22, 6.
- Olive, P. L., Banáth, J. P. (2006). The comet assay: a method to measure DNA damage in individual cells. *Nature protocols*, 1(1), 23.
- Razeghi, O., Solís-Lemus, J. A., Lee, A. W., Karim, R., Corrado, C., Roney, C. H. ve Niederer, S. A. (2020). CemrgApp: An interactive medical imaging application with image processing, computer vision, and machine learning toolkits for cardiovascular research. *SoftwareX*, 12, 100570.
- Rublee, E., Rabaud, V., Konolige, K. ve Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. *2011 International conference on computer vision*, 2564-2571, IEEE.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla A., Bernstein, M., Berg, A.C., Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115, 211-252.
- Sadafi, A., Bordukova, M., Makhro, A., Navab, N., Bogdanova, A. ve Marr, C. (2023). RedTell: an AI tool for interpretable analysis of red blood cell morphology. *Frontiers in Physiology*, 14, 1058720.
- Sanders, J. W., Fletcher, J. R., Frank, S. J., Liu, H. L., Johnson, J. M., Zhou, Z. ve Ma, J. (2019). Deep learning application engine (DLAE): Development and integration of deep learning algorithms in medical imaging. *SoftwareX*, 10, 100347.
- Savitzky, B. H., Zeltmann, S. E., Hughes, L. A., Brown, H. G., Zhao, S., Pelz, P. M., ... ve Ophus, C. (2021). py4DSTEM: A software package for four-dimensional scanning transmission electron microscopy data analysis. *Microscopy and Microanalysis*, 27(4), 712-743.
- Schneider, J., Weiss, R., Ruhe, M., Jung, T., Roggenbuck, D., Stohwasser, R. ve Rödiger, S. (2019). Open source bioimage informatics tools for the analysis of DNA damage and associated biomarkers. *Journal of Laboratory and Precision Medicine*, 4(0), 1-27.

- Spampinato, C., Giordano, D., Di Salvo, R., Chen-Burger, Y. H. J., Fisher, R. B. ve Nadarajan, G. (2010). Automatic fish classification for underwater species behavior understanding. *Proceedings of the first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams*, 45-50
- Temiz, H. (2023). DeepSR: A deep learning tool for image super resolution. *SoftwareX*, 21, 101261.
- Varshni, D., Thakral, K., Agarwal, L., Nijhawan, R. ve Mittal, A. (2019, February). Pneumonia detection using CNN based feature extraction. *2019 IEEE international conference on electrical, computer and communication technologies (ICECCT)*, 1-7, IEEE.
- Vijayalakshmi, A. (2020). Deep learning approach to detect malaria from microscopic images. *Multimedia Tools and Applications*, 79, 15297-15317.
- Wang, B., Kang, Y., Huo, D., Chen, D., Song, W. ve Zhang, F. (2023). Depression signal correlation identification from different EEG channels based on CNN feature extraction. *Psychiatry Research: Neuroimaging*, 328, 111582.
- Wong, P. L. ve Abas, Z. A. (2017). An analysis of human silhouettes with normalised silhouettes images and shape fourier descriptors. *International Journal of Human and Technology Interaction (IJHaTI)*, 1(1), 31-36.
- Wu, S. G., Bao, F. S., Xu, E. Y., Wang, Y. X., Chang, Y. F. ve Xiang, Q. L. (2007, December). A leaf recognition algorithm for plant classification using probabilistic neural network. *2007 IEEE international symposium on signal processing and information technology*, 11-16, IEEE.

DİZİN

A	F	M
arff 31, 33, 35	fourier 1, 2, 4, 6, 7, 14, 15, 16, 40, 41, 42, 67, 68	matlab 31, 33, 34
analiz 3, 10, 19, 47, 61, 68, 69, 70		
B	G	O
bellek 5, 11, 21, 22, 24, 33, 35, 36, 38, 39	geri yayılım 12	octave 34
C	H	P
CeNiN 10, 24, 25, 27, 36, 47, 67, 68, 70	havuzlama 13, 14, 17, 26	paralelleştirme 5, 22, 25, 26, 35, 38, 39, 46, 67
CNNFET 18, 24, 27, 28, 29, 31, 33, 34, 35, 38, 39, 42, 43, 45, 46, 47, 48, 61, 67, 68, 69, 70		
D	İ	S
düzenli ifade 23, 28	implementasyon 3, 4, 5, 6, 18, 24, 36, 37 38, 46, 47, 67	sınıflandırıcı 3, 17, 18, 70
E	K	W
eğitim 4, 5, 10, 11, 16, 17, 18, 19, 20, 27, 29, 42, 44, 47, 58, 62, 64, 66, 68, 69, 70	korelasyon 11, 30, 33, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 63, 64, 69, 70	weka 31, 33, 34, 35, 68



TEKNOVERSİTE



teknoversite **AYRICALIĞINDASINIZ**

İSTE

