

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

**OPTIMIZATION OF DEEP NEURAL NETWORK ARCHITECTURES FOR
THE FOREST FIRE DETECTION**



M.Sc. THESIS

Berrin SAVDA

Department of Electronics and Communication Engineering

Electronics Engineering Programme

JANUARY 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

**OPTIMIZATION OF DEEP NEURAL NETWORK ARCHITECTURES FOR
THE FOREST FIRE DETECTION**



M.Sc. THESIS

**Berrin SAVDA
(504191206)**

Department of Electronics and Communication Engineering

Electronics Engineering Programme

**Thesis Advisor: Prof. Dr. Müştak Erhan YALÇIN
Thesis Co-Advisor: Prof. Dr. Hazım Kemal EKENEL**

JANUARY 2023

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

**ORMAN YANGINI TESPİTİNDE DERİN SİNİR AĞI MİMARİLERİNİN
OPTİMİZASYONU**

YÜKSEK LİSANS TEZİ

**Berrin SAVDA
(504191206)**

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

**Tez Danışmanı: Prof. Dr. Müştak Erhan YALÇIN
Eş Danışman: Prof. Dr. Hazım Kemal EKENEL**

OCAK 2023

Berrin SAVDA, a M.Sc. student of İTÜ Graduate School student ID 504191206, successfully defended the thesis/dissertation entitled “OPTIMIZATION OF DEEP NEURAL NETWORK ARCHITECTURES FOR THE FOREST FIRE DETECTION”, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Prof. Dr. Müştak Erhan YALÇIN**
Istanbul Technical University

Co-advisor : **Prof. Dr. Hazım Kemal EKENEL**
Istanbul Technical University

Jury Members : **Prof. Dr. Ender Mete EKŞİOĞLU**
Istanbul Technical University

.....
Prof. Dr. Cüneyt GÜZELİŞ
Yaşar University

.....
Assoc. Prof. Volkan KILIÇ
İzmir Katip Çelebi University

Date of Submission : 21.12.2022

Date of Defense : 18.01.2023





To my family,



FOREWORD

I'm extremely grateful to my advisor Prof. Dr. Müştak Erhan Yalçın, for his support and patience throughout my thesis and M.Sc. studies. Your guidance on both research as well as on my career has been invaluable.

I would like to express my deepest appreciation to my co-advisor, Prof. Dr. Hazım Kemal Ekenel for his guidance, motivation and continuous support. Also, I am very lucky to have the opportunity to be a member of SiMiT Lab. Our meetings were very inspirational and fun.

I would especially like to thank my amazing family and my friends for their love, support, and constant encouragement. It would have been impossible to finish my studies without their unwavering support.

I want to thank the Scientific and Technical Research Council of Turkey (TUBITAK) for supporting my thesis under ARDEB 1001 Grant No 121E408.

Last but not least, I would like to express my special thanks to Furkan Elmaz. I could not have undertaken this journey without you.

December 2022

Berrin SAVDA



TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ix
TABLE OF CONTENTS	xi
ABBREVIATIONS	xiii
LIST OF TABLES	xv
LIST OF FIGURES	xvii
SUMMARY	xix
ÖZET	xxi
1. INTRODUCTION	1
1.1 Purpose of Thesis	1
1.2 Literature Review	2
1.3 Research Objective and Thesis Outline	2
2. IMAGE CLASSIFICATION	9
2.1 Data Acquisition	9
2.1.1 Collection of dataset	10
2.1.2 Image preprocessing	11
2.2 Deep Learning Methods	12
2.2.1 Convolutional neural networks	12
2.2.2 ResNet101	15
2.2.3 DenseNet-169	17
2.2.4 Inceptionv3	18
2.2.5 RegNet-X-32GF	20
2.2.6 Transfer learning	22
2.2.7 Evaluation metrics	23
2.3 Optimization Techniques	24
2.3.1 Quantization	24
2.3.1.1 Affine/Asymmetric quantization	25
2.3.1.2 Symmetric quantization	25
2.3.1.3 Per-tensor and per-channel quantization	26
2.3.2 Pruning	28
2.4 Results and Discussion	31
3. OBJECT DETECTION	43
3.1 Data Acquisition	43
3.2 Object Detection Methods	43
3.2.1 Faster R-CNN	44
3.2.2 YOLOv5	46
3.2.2 Evaluation metrics	48
3.3 Results and Discussion	49
4. CONCLUSIONS AND RECOMMENDATIONS	53
REFERENCES	55
CURRICULUM VITAE	63



ABBREVIATIONS

CNN	: Convolutional Neural Network
UAV	: Unmanned Aerial Vehicle
SVM	: Support Vector Machine
FC	: Fully Connected
PTQ	: Post-training Quantization
QAT	: Quantization-aware Training
AP	: Average Precision
mAP	: Mean Average Precision



LIST OF TABLES

	<u>Page</u>
Table 2.1: Performance evaluation results of vanilla models.	33
Table 2.2: Performance evaluation of per-tensor affine quantization.	36
Table 2.3: Performance evaluation of per-tensor symmetric quantization.	36
Table 2.4: Performance evaluation of per-channel affine quantization	36
Table 2.5: Performance evaluation of per-channel symmetric quantization.....	37
Table 2.6: Original and quantized models comparison.....	38
Table 2.7: Size and inference time of non-quantized models.	38
Table 2.8: Size and inference time results of per-tensor affine quantization.	39
Table 2.9: Size and inference time results of per-tensor symmetric quantization. ...	39
Table 2.10: Size and inference time results of per-channel affine quantization.	40
Table 2.11: Size and inference time results of per-channel symmetric quantization.	40
Table 2.12: Performance evaluation results of quantized and pruned models.....	41
Table 3.1: Vanilla models evaluation results	51
Table 3.2: Size and inference time of models	51
Table 3.3: Performance evaluation of optimized models.....	52
Table 3.4: Memory and inference time of optimized models	52



LIST OF FIGURES

	<u>Page</u>
Figure 2.1: The incident dataset composition.	10
Figure 2.2: The incident dataset image samples.	10
Figure 2.3: Class-positive image samples (fire).....	11
Figure 2.4: Class-negative image samples (no-fire).	11
Figure 2.5: CNN Structure.	13
Figure 2.6: Convolution example.....	13
Figure 2.7: Pooling examples.	14
Figure 2.8: Example of FC layer architecture.....	14
Figure 2.9: Activation function examples.....	15
Figure 2.10: Residual block representation taken from [42].	16
Figure 2.11: ResNet architecture taken from [41].	17
Figure 2.12: A 5-layer dense blok taken from [43].....	18
Figure 2.13: Naive inception module.....	19
Figure 2.14: Dimension-reduced inception modules.	19
Figure 2.15: Inception architecture taken from [50].	20
Figure 2.16: AnyNet architecture summary taken from [51].....	20
Figure 2.17: RegNet-X representation taken from [51].	21
Figure 2.18: RegNet-X-32GF setup taken from [51].	22
Figure 2.19: Transfer learning block diagram.	23
Figure 2.20: Per-tensor and per-channel quantization scheme representation.....	26
Figure 2.21: QAT configuration graph taken from [54].	27
Figure 2.22: PTQ pipeline on the right and QAT pipeline on the left.	28
Figure 2.23: Pruning methods summary.	29
Figure 2.24: Unstructured pruning representation.	29
Figure 2.25: Pruning implementation pipeline.	30
Figure 2.26: Training and validation loss graphs for ResNet101 (a), RegNet-X-32GF (b), Inceptionv3 (c), DenseNet-169 (d).....	32
Figure 2.27: Pruning ratio vs accuracy graphs for ResNet101 (a), RegNet-X-32GF (b), Inceptionv3 (c), DenseNet-169 (d).....	34
Figure 2.28: Pruning ratio vs memory size graphs for ResNet101 (a), RegNet-X-32GF (b), Inceptionv3 (c), DenseNet-169 (d).....	35
Figure 3.1: Image samples from object detection dataset.....	43
Figure 3.2: Object detection methods summary.	44
Figure 3.3: Faster R-CNN Pipeline.....	45
Figure 3.4: Faster R-CNN detection pipeline taken from [65]	45
Figure 3.5: YOLOv5 grid method example.	46
Figure 3.6: Bounding box representation.....	47
Figure 3.7: IOU calculation.	47
Figure 3.8: Object detection mAP score calculation steps.....	48
Figure 3.9: (a) Original image, (b) Horizontal flip of (a), (c) -15 rotated version of (a), (d) +15 rotated version of (a).....	49

Figure 3.10: (a) Faster R-CNN training and validation loss graph, (b) YOLO5 training and validation graph.....	50
Figure 3.11: Faster R-CNN precision, recall, F1 and PR graphs.....	50
Figure 3.12: YOLOv5 precision, recall, F1 and PR graphs	51



OPTIMIZATION OF DEEP NEURAL NETWORK ARCHITECTURES FOR THE FOREST FIRE DETECTION

SUMMARY

Forests are critical for all living creatures in vital issues such as biodiversity conservation, the sustainability of the ecosystem, and the continuation of the water and nutrient cycle. Forest fires are large uncontrolled incidents that burn and quickly spread through wild landscapes. According to research, average temperatures are on the rise forest fires seem to appear more often and more destructive. Therefore, detecting forest fires as early as possible and responding quickly have become essential to toward potentially deadly disasters. Various studies are conducted in the literature on the detection of forest fires. Besides more traditional image processing and machine learning methods, deep learning-based methods have been frequently employed in the recent literature. Despite their exceptional performances, it has been a challenge to utilize resulting prediction models in edge and low-power computational devices due to their massive sizes and computational complexity. However, there are no studies in the literature addressing these issues for detecting forest fires. On the other hand, there are well-established optimization techniques to make deep neural networks less power-hungry and more computationally efficient. This thesis aims to bridge this gap by utilizing such optimization techniques for their use in deep learning-based forest fire detection models.

In this thesis, we aim to develop efficient as well as accurate deep learning models for detecting forest fires. Firstly, a dataset is created by using the public data in the literature. Then, ResNet101, RegNet-32X-GF, DenseNet-169 and Inceptionv3 networks were trained to detect forest fires. All methods had comparable and satisfactory performance reaching over 95% test accuracy. Then, the resulting models are optimized and compared using various quantization and pruning methods. DenseNet showed better resiliency against accuracy drop and size reduction compared to the other methods. Furthermore, the classification problem extended to object detection to explore further the potential of the proposed approach. Faster R-CNN was employed with the DenseNet backbone and optimized with previously determined optimal hyperparameters. Then, it was compared to the state-of-the-art optimized YOLOv5 and showed not only higher accuracy but also lower size. Therefore, this study shows that the employed methods and followed methodology has a promising potential to bring the power of deep learning-based techniques to the edge and low-power devices for the forest fire detection problem.



ORMAN YANGINI TESPİTİNDE DERİN SİNİR AĞI MİMARİLERİNİN OPTİMİZASYONU

ÖZET

Biyolojik çeşitliliğin korunması, ekosistemin sürdürülebilirliği, su ve besin döngüsünün devamı gibi hayati konularda tüm canlılar için ormanlar kritik öneme sahiptir. Orman yangınları, yaban arazilerde hızla yayılarak yanan büyük kontrolsüz olaylardır. Son yıllarda, küresel ısınmanın etkisiyle dünyamızda iklim değişiklikleri gözlenmektedir. Ortalama sıcaklıkların artması yangına temel oluşturan ve yangın mevsimi olarak adlandırılan hava koşullarının uzun dönemler görülmesine neden olmuştur. Bu nedenle orman yangınları daha sık ve daha zor müdahale edilebilen bir afet türü olmuştur. Bu yangınlardan doğabilecek hasarları ve ölümcül sonuçları önlemek için erken ve doğru tespit yapmak hayatı bir öneme sahiptir. Bu şekilde yangın söndürme çalışmalarına daha hızlı başlanabilir ve yaşam alanlarını korumak için gerekli önlemler zamanında alınabilir. Bu doğrultuda orman yangınlarının tespiti araştırmacılar için ilgi çekici bir konu olmuştur ve literatürde çeşitli çalışmalar bulunmaktadır. Geleneksel görüntü işleme ve makine öğrenmesi yöntemlerinin sıklıkla kullanıldığı görülmüştür. Bu çalışmalar görüntülerden istatistiksel metotlar kullanarak özellik çıkarımı yapar ve daha sonra uygun makine öğrenmesi yöntemi seçilerek görüntüde yangın olup olmadığını tespit eder. Bu çalışmaların çıktıları ışığında orman yangını tespitinde görüntüde bulunan bazı önemli noktalar belirlenmiştir. Örneğin, yangın sırasında ortaya çıkan alev ve dumanların kendine has renk, şekil ve hareket yapısı vardır. Tespit esnasında bunların doğru belirlenmesi modelin performansını doğrudan etkilemektedir. Geleneksel görüntü işleme ve makine öğrenimi yöntemlerinin yanı sıra, son zamanlarda literatürde derin öğrenmeye dayalı yöntemler de sıklıkla kullanılmaktadır. Derin öğrenme yöntemleri özellik çıkarımı kısmını ek veriye gerek olmadan sadece görüntüleri kullanarak yapar ve hem bahsedilen özellikleri kolaylıkla ayırt ederken aynı zamanda daha detaylı özelliklerin çıkarımını da yapar. Bu nedenle özellikle görüntü ve video tabanlı uygulamalar için oldukça avantajlıdır. Literatürde bulunan derin öğrenme tabanlı çalışmalar genellikle görüntü sınıflandırma ve nesne tespiti olmak üzere iki farklı yaklaşımla ele alınmıştır. Görüntü sınıflandırma problemlerinde görüntüde yangın olup olmadığına bakılırken nesne tespitinde aynı zamanda yangın olan bölgenin konum tespiti de yapılır. Bu çalışmalarda kullanılan yöntemler ve çalışmanın odak noktası model başarımı üzerine olmuştur. Olağanüstü performanslarına rağmen, ortaya çıkan tahmin modellerini uç ve düşük güçlü hesaplama cihazlarında kullanmak, devasa boyutları ve hesaplama karmaşıklığı nedeniyle zorlayıcıdır. Öte yandan, derin sinir ağlarını daha az güç ile çalışan ve hesaplama açısından daha verimli hale getirmek için kullanılan başarılı optimizasyon teknikleri vardır. Literatürde orman yangınlarının tespiti için bu konuları ele alan herhangi bir çalışma bulunmamaktadır. Bu tez, derin öğrenme tabanlı orman yangını algılama modellerinde kullanımları için bu tür optimizasyon tekniklerini kullanarak literatürdeki bu boşluğu doldurmayı amaçlamaktadır.

Derin öğrenme tabanlı yapılan bilgisayarlı görü uygulamalarının başarılı olmasının ana nedenlerinden biri kullanılan algoritmaların sahip olduğu kompleks yapılardır. Bu yapılar yüksek başarımlar sağlarken aynı zamanda hafızada fazla yer tutmaları, donanımlar üzerinde ısınma problemine yol açmaları, yüksek enerji gereksiniminden dolayı yavaş çalışmaları gibi dezavantajları vardır. Bu dezavantajlar kaynak kısıtlaması olan uygulamalarda daha çok gündeme gelmektedir. Özellikle düşük güçle çalışan cihazlar kullanılarak yapılan uygulamalarda bu gibi engellerle sıklıkla karşılaşılır. Bunun için literatürde çeşitli metotlar bulunmaktadır. Örneğin daha güçlü donanımların tercih edilmesi, kullanılan modellerin değiştirilmesi ya da model optimizasyonu gibi. Bu uygulamalardan en verimli olanı model optimizasyonu tekniklerinin kullanılmasıdır. Bunun nedeni hali hazırda üzerine çalışılmış modelleri ve düzeneği değiştirmeden bazı matematiksel yöntemler kullanılarak var olan modelleri olduğundan daha küçük ve daha hızlı hale getirmeyi sağlamasıdır. Bizim problemimiz özelinde yapılan araştırmalar ve deneyler sonucunda probleme en uygun olacak optimizasyon yöntemleri nicemleme ve budama olarak belirlenmiştir. Nicemleme kullanılan sayı verilerini daha küçük veri tiplerine dönüştürülerek daha hızlı işlemler yapılmasını ve daha az hafıza kullanılabilmesini sağlar. Örneğin bizim problemimizde sayı değerleri 32-bitlik yer kaplayan kayan noktalı sayılar yani float32 ile gösterilir. Float32'den 8-bitlik yer kaplayan int8 formatına geçildiğinde modelin yaklaşık olarak dört kat daha az yer kaplaması beklenir. Bu işlem yapılırken farklı yaklaşımlar mevcuttur. Nicemleme eğitim sırasında ya da eğitim sonrası uygulanabilir. Yapılan uygulamalara göre bu tercih değişebilir fakat yapılan çalışmalar incelendiğinde eğer karmaşık ve çok katmanlı bir model üzerinde çalışılıyorsa eğitim sırasında nicemleme yapmak çok maliyetli olacaktır. Bu nedenle eğitim sonrası nicemleme kullanılması bizim çalışmamız için daha avantajlı olduğu görülmüştür. Kolay uygulanabilir olmasının bazı dezavantajları da bulunur. Örneğin doğru yöntemler belirlenmediğinde modelin başarımında büyük kayıplar gözlemlenmesi mümkündür. Bunu önlemek için eğitilmiş modeli bazı adımlardan geçirmek ve uygulanacak nicemleme yöntemini iyi seçmek gerekir. Nicemlemede sayı değerlerinde geçiş yaparken kullanılan iki önemli yöntem vardır. Bu geçiş simetrik ya da asimetrik olarak yapılabilir. İki yöntem arasındaki en önemli fark sıfır değerinin nasıl gösterileceğinden gelir. Simetrik nicemleme sıfır sıfır olarak kullanarak kalan değerleri bu eksenle simetrik olarak dağıtmaya çalışır. Fakat asimetrik nicemlemede sıfır noktası olarak adlandırılan bir değişken daha vardır ve sıfırın gösterimi bu değişken hesaplanarak yapılır. Sıfır noktası hesaplanırken elde bulunan parametreler kullanıldığından daha gerçeği yansıtan bir geçiş sağlanmış olur. Simetrik nicemlemeyi kullanmak daha az matematiksel işlem gerektirdiğinden daha hızlı olsa da bahsedilen nedenden dolayı model başarımında düşüşe neden olabilir. Aynı zamanda nicemleme yapılırken bu değişimde belirlenecek nicemleme parametreleri farklı şekillerde oluşturulabilirler. Tensör başına nicemleme parametreleri hesaplanabilir ya da her kanal için ayrı ayrı bu parametreler belirlenebilirler. Teoride baktığımızda nicemleme yöntemlerinin bazılarının daha iyi sonuç vereceğini öngörebiliriz. Örneğin bahsedilen bilgiler ışığında kanal başına asimetrik nicemleme kullandığımızda modeli en az kayıpla optimize edeceği beklenen bir sonuçtur. Fakat teoride bunlar beklenen sonuçlar olmasına karşın pratikte farklı sonuçların elde edildiği gözlemlenmiştir.

Diğer bir optimizasyon yöntemi olan budama ağdaki gerekli olmayan bağlantıları bulmayı amaçlar. Bu yöntemin başarılı olmasının altındaki mantık piyango bileti hipotezine dayanır. Bu hipotez çok katmanlı ağı yakaladığı başarımları doğru başlangıç değerleri ile başlatıldığında az katmanlı ağlarında aynı oranda yakalayacağını belirtir.

Bu nedenle karmaşık ve çok katmanlı ağlardan gereksiz olan bağlantıları veya yapıların çıkarılması model başarımını en minimum etkide etkilemesi beklenmektedir. Bu şekilde modellere budama yöntemi farklı şekillerde uygulanabilir. Budamada belirlenmesi gereken en önemli nokta ağdaki hangi bağlantıların çıkarılacağıdır. Buna karar verirken bazı matematiksel hesaplamalar yapılabildiği için rastgele olarak da belirlenebilir. Matematiksel hesaplamaların kullanıldığı yaklaşımda budanacak bağlantılar tensör ya da kanal içindeki değerlerin L1, L2 gibi metrikler kullanılarak uzaklık hesabına göre bakılmaktadır. Burada L1 uzaklık metriğini kullanmak sıklığı arttırdığından daha etkili sonuçlar vermektedir. Diğer bir önemli olan nokta ise ağdan çıkarılacak yapıları belirlerken önemsiz görülen bağlantılar bireysel olarak mı çıkarılacaktır yoksa tüm bir yapı incelenerek yapısal çıkarım mı yapılacaktır. Bireysel bağlantıların kesildiği yöntem yapılandırılmamış budama ve yapıların ağdan çıkarıldığı yöntem yapılandırılmış budama olarak adlandırılır. Burada iki yöntemin de avantajları ve dezavantajları vardır fakat yapılan çalışmalarda L1 yapılandırılmamış budamanın model performansına en az zarar veren şema olduğu görülmüştür.

Bu tezde, bahsedilen yöntemler kullanılarak orman yangınlarını tespit etmek için verimli ve doğru derin öğrenme modelleri geliştirmeyi amaçlanmıştır. İlk olarak, literatürdeki açık veriler kullanılarak yeni bir veri kümesi oluşturulmuştur. Burada orman yangını problemine uygun görüntüler toplanırken geçmiş tecrübelerden faydalanılmıştır. Özellikle geleneksel özellik çıkarımı yapılarak uygulanan çalışmaların sonucunda görüntüde önemli olduğu belirlenmiş noktalara dikkat ederek görüntüler toplanmıştır. Daha sonra ResNet101, RegNet-32X-GF, DenseNet-169 ve Inceptionv3 ağları orman yangınlarını algılamak üzere eğitilmiştir. Tüm yöntemler, %95'in üzerinde test doğruluğuna ulaşmış ve tatmin edici performansa sahip olduğu görülmüştür. Yapılan test sonuçlarında aşırı öğrenme, eksik öğrenme veya verilerde dengesizlik problemleri ile karşılaşılmamıştır. Ardından, ortaya çıkan modeller optimize edilmiş ve çeşitli nicemleme ve budama yöntemleri kullanılarak karşılaştırılmıştır. Farklı şemalarda uygulanan nicemleme ve budama yöntemlerinin modeller üzerinde farklı etkileri olduğu görülmüştür. Her model için uyumlu olan optimizasyon yöntemi, yapının özelliklerine göre farklılık göstermiştir. Yapılan bu deneyler sonucunda optimizasyon adımı farklı yöntemlerin kıyaslanması modelin en az kayıpla optimize edilebilmesi için doğru metotları bulmada faydalı bir yol olduğu gözlemlenmiştir. Deney sonuçları incelendiğinde DenseNet-169, diğer modellere kıyasla performansta daha az kayıp göstererek daha dayanıklı kalmış ve boyutta daha fazla düşüş göstermiştir. Önerilen yaklaşımın potansiyelini daha fazla araştırmak için optimize edilen modellerden en iyi sonuç veren mimari nesne tespiti yöntemlerinde de uygulanmıştır. Bunun için, Faster R-CNN, DenseNet-169 omurgası ile kullanılmış ve önceden belirlenmiş optimum hiperparametrelerle optimize edilmiştir. Daha sonra literatürde çokça kullanılan optimize edilmiş YOLOv5 ile karşılaştırıldı. Bu karşılaştırma sonucunda optimize edilmiş Faster R-CNN yalnızca daha yüksek doğruluk oranına ulaşmakla kalmamış aynı zamanda daha düşük boyutlara sıkıştırılarak kullanımı daha avantajlı hale gelmiştir. Ek olarak hızlı tahminleme yönünden güçlü bir mimari olan YOLOv5'e hız anlamında yaklaşarak bu konuda da gelişme göstermiştir. Sonuçlar incelendiğinde, bu çalışmada kullanılan yöntemlerin ve izlenen metodolojinin, orman yangını tespiti problemi için derin öğrenmeye dayalı tekniklerin gücünü sınır ve düşük güçlü cihazlara getirme konusunda umut verici bir potansiyele sahip olduğunu göstermiştir. İyolojik çeşitliliğin korunması, ekosistemin sürdürülebilirliği, su ve besin döngüsünün devamı gibi hayati konularda tüm canlılar için ormanlar kritik öneme sahiptir. Orman

yangınları, yaban arazilerde hızla yayılarak yanan büyük kontrolsüz olaylardır. Yapılan araştırmalara göre, ortalama sıcaklıkların artışıyla orman yangınları daha sık görülmeye başlanmış ve daha yıkıcı hale gelmiştir. Bu nedenle, orman yangınlarını mümkün olduğu kadar erken tespit etmek ve hızlı bir şekilde müdahale etmek, ölümcül sonuçların önlemesi için oldukça gereklidir. Orman yangınlarının tespiti ile ilgili literatürde çeşitli çalışmalar yapılmaktadır. Geleneksel görüntü işleme ve makine öğrenimi yöntemlerinin yanı sıra, son zamanlarda literatürde derin öğrenmeye dayalı yöntemler de sıklıkla kullanılmaktadır. Olağanüstü performanslarına rağmen, ortaya çıkan tahmin modellerini uç ve düşük güçlü hesaplama cihazlarında kullanmak, devasa boyutları ve hesaplama karmaşıklığı nedeniyle zorlayıcıdır. Literatürde orman yangınlarının tespiti için bu konuları ele alan herhangi bir çalışma bulunmamaktadır. Öte yandan, derin sinir ağlarını daha az güç ile çalışan ve hesaplama açısından daha verimli hale getirmek için kullanılan başarılı optimizasyon teknikleri vardır. Bu tez, derin öğrenme tabanlı orman yangını algılama modellerinde kullanımları için bu tür optimizasyon tekniklerini kullanarak literatürdeki bu boşluğu doldurmayı amaçlamaktadır.

Bu tezde, orman yangınlarını tespit etmek için verimli ve doğru derin öğrenme modelleri geliştirmeyi amaçlanmıştır. İlk olarak, literatürdeki açık veriler kullanılarak yeni bir veri seti oluşturulmuştur. Daha sonra ResNet101, RegNet-32X-GF, DenseNet-169 ve Inceptionv3 ağları orman yangınlarını algılamak üzere eğitilmiştir. Tüm yöntemler, %95'in üzerinde test doğruluğuna ulaşmış karşılaştırılabilir ve tatmin edici performansa sahip olduğu görülmüştür. Ardından, ortaya çıkan modeller optimize edilmiş ve çeşitli nicemleme ve budama yöntemleri kullanılarak karşılaştırılmıştır. DenseNet-169, diğer yöntemlere kıyasla doğruluk düşüşüne daha dayanıklı kalmış ve boyutta daha fazla düşüş göstermiştir. Ayrıca sınıflandırma problemi, önerilen yaklaşımın potansiyelini daha fazla araştırmak için nesne tespiti yöntemlerinde de uygulanmıştır. Faster R-CNN, DenseNet-169 omurgası ile kullanıldı ve önceden belirlenmiş optimum hiperparametrelerle optimize edilmiştir. Daha sonra tekniğin bilinen durumu olan optimize edilmiş YOLOv5 ile karşılaştırıldı ve yalnızca daha yüksek doğruluk değil, aynı zamanda daha düşük boyut da gösterdi. Bu nedenle, bu çalışma, kullanılan yöntemlerin ve izlenen metodolojinin, orman yangını algılama sorunu için derin öğrenmeye dayalı tekniklerin gücünü sınır ve düşük güçlü cihazlara getirme konusunda umut verici bir potansiyele sahip olduğunu göstermektedir.

1. INTRODUCTION

1.1 Purpose of Thesis

Forests, which have been badly affected by global warming, climate change, urbanization, and the opening up of agricultural areas, face an increasing threat of forest fires. Forest fires often occur in weather conditions such as high temperatures, low relative humidity, and strong winds and are challenging to take under control. Nowadays, global warming has increased the frequency of occurrence of conditions that will cause fires by lead an increase in average temperature, atmospheric instability, and the occurrence of droughts. While they can start naturally, forest fires are often caused by human activity. Factors such as campfires that become uncontrollable, improperly handled cigarettes, arson, and glass shards are the most common causes of forest fires [1]. The motivation for preventing forest fires is forests have an essential role in vital issues such as preventing erosion, maintaining the balance of weather temperature, cleaning to air, the continuation of the water and nutrient cycle, etc. Since forests are home to many living things, when a forest fire occurs, animal and human feeding and shelter areas will be damaged, threatening nature and biodiversity, and some animal species may be endangered. At the same time, the emergence of infectious diseases in recent years has caused an increase in chronic lung diseases in people. The research predicts that this situation will negatively affect people with lung diseases, as forest fires lead to a gradual decrease in oxygen resources and clean air in the world [2]. Therefore, the importance of forests has become more critical than ever before. Recent years have seen massive forest fires worldwide that have killed over a hundred people and left thousands homeless. Countries around the world are dealing with devastating forest fires. On average, forest fires burn up to millions of acres of land annually. Areas with warmer climates such as Siberia, California, Australia, Turkey, Greece, Italy, France, and Algeria frequently struggle with forest fires. The presence of extremely high temperatures worldwide in 2021 and 2022 has led to an active fire season. For example, the Dixie fires in California started due to a lightning strike in dry weather and could not be controlled for about two and a half months. In

2021 Turkey forest fires, 8 people lost their lives in forest fires, most of which broke out in 53 provinces in the Mediterranean, Aegean, Marmara, Western Black Sea, and Southeastern Anatolia Regions, while more than 150 thousand hectares of forest area and residential area were turned into ashes with a significant increase compared to previous years. It is expected that forest fires will occur more often and severely in the coming years with the effect of global warming. For these reasons, forest fire detection has become an important research topic. Therefore, this thesis aims to prevent damage caused by forest fires with efficient computer vision techniques.

1.2 Literature Review

There are many approaches to detecting forest fires, such as satellite systems, wireless sensor networks, image processing, color and moment mixed techniques, machine learning, etc. Sensor-based approaches are commonly used in fire detection [3–5], but these methods are ineffective in open spaces such as forests. Besides using sensor-based approaches, with the development of image processing techniques, studies have begun using images and videos to detect fire and reduce the rate of false alarms. Detection of forest fires with image processing techniques is implemented by monitoring the forest areas and collecting images, then processing these images using various statistical methods. In order to detect fire areas in the images, several methods exist in the literature. The essential elements in fire detection are determining the flames and smoke region presented in the images. While choosing these elements, the images can be gray-scale or colored. The shape of the fire and changes in color intensities are indicative features in detections made using gray-scale images. Color intensities differ according to the temperature of the object. High temperatures appear in white, while low temperatures appear in darker tones [6]. Estimates will not be accurate when there are objects with the same color intensity as the fire in the environment. Hence, using gray-scale images to determine the fire by only examining the intensities and shape does not yield reliable results. Thus, using color images is a more proper approach to understanding the characteristic features of the fire. Flames and smoke have their unique color and texture. Flame usually has a reddish hue. It is also seen in different colors depending on the fuel that feeds the fire and its temperature. Since the primary fuel material in forest fires will be wood, the color of the flame shows transitions from red to yellow. Considering these color features, there

are various studies in the literature. Chen et al. suggested using a two-stage method using video images to detect the fire [6]. It has been represented that there is a possibility of fire in the ranges where the hue value is 0-60. They mentioned that only examining hue values are not given reliable results; the RGB values are also calculated to determine the areas with the possibility of fire. The second stage has been developed to prevent false alarms. To justify fire areas and prevent false alarms, comparisons are made on video frames by extracting differences. Thus, examining these differences determined whether the spreading behavior in the image was the same as during the fire, and it was justified that the regions were fire areas. In another study by Chen et al. using the RGB color model, both chromatic and dynamic analyses of the image were made. They aimed to obtain more information from the image and make it easier to distinguish when there are objects with similar colors to the fire. There are also studies on different color models other than RGB in the literature. Horng created a model that predicts both the detection of the fire region and the temperature of the flames by using the HSI color model in the method proposed [7]. This method determines the areas with the possibility of fire according to the HSI values. The area that does not include fire is extracted from the image and using remain part, the temperature of the flame is estimated. Temperature information has been added to eliminate objects with fire-like colors and properties. It has been determined whether there is a fire in the environment by examining the image features and reinforcing the temperature values. Marbach et al. created a detection method according to fire intensity using the YUV color space [8]. In this method, they examined the flickering movement by looking at the frequency range of the fire and extracted the fire zone using temporal features. Shape analysis is as important as distinguishing the colors of the fire in the image. In the study by Liu and Ahuja [9] they discuss that the methods previously used in static shape analysis will be insufficient for detecting fire that behaves stochastically. Therefore, they presented that the spectral, spatial and temporal model in the image is much more informative in representing the movements of the fire region. As the studies introduce, determining the color and motion characteristics and correctly analyzing them are important factors in detecting fire. For this reason, methods that examine two features simultaneously provide relatively more reliable results. Dedeoğlu et al., in their study, presented a method that uses both these features [10]. They determined the color change in moving regions using temporal variation information, color, and spatial wavelet transform. In addition to color properties,

wavelet and object boundary analyses were performed to reduce the occurrence of false alarms. Töreyn et al. [11] suggested an algorithm that combines color characteristics and motion properties, considering the fire and smoke. The moving regions were marked in video sequences, and color analysis of these regions was made. Then the entropies of the wavelet domain energies of these properties were calculated, and whether there was a fire in the image was decided. Zhang et al. [12] used wavelet and Fast Fourier Transform-based methods by combining them. They extracted an outline of the fire region with the Otsu method, and FFT checks if the outline fluctuates with the same frequency with flames. The wavelet method is used to analyze these extracted outlines. Qi and Ebert [13] proposed a method that uses color and motion features to analyze the temporal variation of fire intensity. They used RGB color space to extract pigmentation values and HSV for intensity properties. By using these properties, they predicted fire areas from video clips. Chi et al. proposed a method combining chromatic, dynamic, texture, and contour features [14]. They extracted moving regions with the gradient motion history image method and used the K-Means algorithm to determine fire pixels in the image sequences. To make a final decision, they applied local binary pattern and support vector machine (SVM) methods. Kong et al. [15] suggested using logistic regression and temporal smoothing methods to make a fast fire detection in their study. By combining size, color and motion features with logistic regression, the fire zone was removed and aimed to reduce false alarms with temporal smoothing. When monitoring forests, flames usually do not appear in the image. Therefore, it is vital to identify the smoke in the image. The smoke can be light or dark in color, depending on the fuel type. They are usually lighter colors in forest fires. In cases where they are dark in color, it is observed that the fire that starts in the forest is formed by feeding from different fuels near the residential areas. Gomez et al. [16] used optical flow techniques and wavelet analysis to detect smoke in open spaces. The smoke shows chaotic movement, and it is presented that this movement can be seen with the optical flow technique. In this regard, Töreyn et al. [17] developed a method for smoke detection in the video by considering the frequency characteristics of smoke. The image estimates the background, and regions of decreasing high frequencies are observed using the spatial wavelet transform. At the same time, the chrominance values of the pixels are checked to distinguish the gray color of the smoke. As a result of these analyses, areas with smoke are determined. Chen et al. [18] proposed a method to determine smoke in forest fires by using a

chromaticity-based static decision rule, which analyzes the color of smoke, and a dynamic decision rule, which examines the propagation properties of smoke. Yang et al. presented that in addition to the features extraction method in the literature, changing irregularities of contour and the changing unevenness of smoke density is essential. They have created a detection mechanism that decides whether there is smoke or not using SVM [19]. In the method presented by Ma et al. [20], by considering the movement of the smoke, the moving regions in the image were determined using the Kalman filter and moving history image (MHI) methods together. Using the offline trained Gaussian mixture model in RGB color space, the changes in gray color were observed, and the smoke was detected. Tung and Kim used the approximate median method, fuzzy c-means (FCM), temporal and spatial features to extract the area of smoke movement in their study [21]. They proposed a detection method by using these extracted features with SVM training. As mentioned, the color, shape, and movement structure of flame and smoke are distinctive features in detecting forest fires. It is observed that the success of the model increases when more meaningful features are extracted from the images and videos. When determining forest fire, some approaches only detect flames or smoke. However, the best practice is to establish powerful algorithms to examine both elements together.

Traditional image processing methods require a feature extraction step, and finding all meaningful information from images with this can be not efficient. Deep learning methods, which have not lost their popularity today, are often used to reach and analyze the most useful information on the image. Kim et al. [22] present a fire detection method using deep learning algorithms with aerial images. For this purpose, they created a CNN architecture with three convolutional layers, three pooling layers, and two fully connected layers. The dropout method was used to prevent overfitting and reduce errors. The training was performed using mini-batch gradient descent and stochastic gradient descent. At the end of this training, 81.46% accuracy was achieved by making the binary classification. Popular deep learning models, created to deal with large datasets such as the ImageNet dataset [23], have also been used in forest fire detection problems with transfer learning. Gonzalez et al. [24] developed an AlexNet-based architecture for fire detection. The proposed architecture is named Simple Feature Extraction with Fully Convolutional Network AlexNet, Single Deconvolution (SFEwAN-SD). The first part of the model is based on AlexNet for finding fire regions

from images, but it causes a loss in resolution. The second part of the architecture focuses on extracting color and texture features with a handle to resolve the problems. Performance of the proposed model compared with the study of Rossi et al. [25] and accuracy increased by 4%. As a similar approach, Hüttner et al. used Inceptionv3 with transfer learning for fire detection in their work [26]. Different learning rates and optimization functions were tested to find the best accuracy in avoiding overfitting problems. They performed 4000 iterations with a 0.1 learning rate, gradient descent as an optimization function, and reduced mean for the reduction function. They present their results compared with [27], [28] and achieve 99.4% accuracy.

In recent years object detection algorithms have become more popular due to the use of UAVs in forest fire detection problems. Shen et al. [29] have approached fire detection as an object detection problem. They proposed to use of a robust object detection algorithm called You Only Look Once (YOLO) for real-time fire detection. They used a simplified version of YOLO with twelve convolution layers instead of twenty-four convolutional layers. The model was trained on a relatively shallow dataset which consists of 172 images with fire flames. Authors have used data augmentation methods to reproduce the dataset, and with augmentation, they used 10,320 training images. As a result, they present 76% accuracy of their model. As another object detection algorithm, Faster R-CNN was used to detect forest fire detection by Zhang et. al [30]. In their study, they present that the dataset used in forest fire detection was not efficient for robust detection. Therefore, to diversify the dataset, they created 1,000 differently shaped synthetic smoke with a green screen and added it to 12,620 forest-like backgrounds in these smoke images. Faster R-CNN trained with created dataset, and they achieved a 99.75% detection rate. In addition, according to their test results on real videos, the model did not has identical performance scores as test results on synthetic data. It can be concluded that the usage of synthetic data cannot always be reliable [31]. There are various methods in the literature to execute the mentioned methods more efficiently. Especially in applications made on low-power systems, such as forest fire detection, it is advantageous to reduce the cost and increase the efficiency of the models. In this manner, researchers have focused on model optimization techniques in recent years. There are many optimization techniques used in the literature. The most commonly used methods are quantization, pruning, filter compression, matrix factorization, network architecture search, and knowledge

distillation [32]. In this study, quantization and pruning, which are the most popular optimization methods, were applied. Quantization and pruning are implemented in different ways with different configurations [33], [34]. Therefore, it is important to determine the most suitable method for the model and the problem.

1.3 Research Objective and Thesis Outline

Although deep learning methods used in the literature for forest fire detection have shown very successful results, these costly methods encounter limitations such as memory space, processor power and battery life in low-power devices. For example, unmanned aerial vehicles (UAVs) are often used when monitoring large areas such as forests, and these vehicles work with low-power devices. Since the methods used in the literature consume a lot of power and memory space when working on such devices [35], it is challenging to employ them with maximum efficiency. In order to use models on low-power devices, different optimization techniques made models faster, smaller and more efficient with minimum loss in their performance. Although model optimization is a beneficial method, it is rarely used in applications. To the best of our knowledge, there are no studies focused on optimizing neural networks for forest fire detection problems. This shortcoming in the literature constitutes the main research topic of this thesis. It has been studied to optimize the complex and costly neural networks used in forest fire detection without affecting their predictive capabilities. Quantization and pruning were determined and applied as the most suitable optimization methods for the models. This study has been prepared in several chapters in order to use these methods for forest fire detection, to evaluate their suitability and performance.

Chapter 2 proposed the use of optimized image classification methods for forest detection. For this purpose, a new large data set was created, extracting forest fire images from the Incident and VIDI public datasets. Using this dataset, ResNet101, RegNet-X-32GF, Inceptionv3, and DenseNet-169 architectures were trained. Then, these trained networks are optimized by applying L1 unstructured, random structured, L1 structured, random structured pruning, and per-tensor affine, per-tensor symmetric, per-channel affine, per-channel symmetric quantization individually. By examining

the results, the performances of the models were compared, and the most suitable methods were determined.

Chapter 3 explains the suitability of optimized object detection methods in forest fire detection. A new dataset that had not been used in the literature before was created by gathering aerial images from forest fires. Considering the outputs of Chapter 2, the performance of the optimized models in the object detection problem was observed. Therefore, the most resilient architecture for optimization in forest fire detection was replaced with Faster R-CNN's backbone. Then optimized Faster R-CNN compared to state-of-the-art optimized YOLOv5.



2. IMAGE CLASSIFICATION

2.1 Data Acquisition

2.1.1 Collection of the image dataset

In studies on forest fire detection, frequently used datasets consist of mostly close-up shots of forest fire images or fire images regardless of outdoor and indoor space. To give examples of the datasets used in the literature, the MIVIA dataset [36] consists of 14 fire and 17 no-fire video images. Fire-free images have similar structures to the fire and smoke shapes like red moving objects, clouds, etc. which is a distinguishing feature for ascertaining the accuracy of the model. Corsican Fire Dataset is used for fire detection applications that have 2000 images [37]. A dataset was prepared by gathering images from sources on the internet, firefighters, and researchers. RF Dataset contains 12620 synthetic and real smoke images. Synthetic images are produced using a smoke frame with a green background added to different forest images [30]. In real-time applications, forests are monitored distantly with generally low-resolution cameras. Thus, focusing on only fire images or only smoke images can be deceptive for the determined forest fires. Training the model with the same type of images can lead the model to make inferences beyond memorization and this causes unreliable predictions, especially in real-time applications. It is necessary to diversify the images in the dataset and examples of possible scenarios. It is necessary to diversify the images in the dataset and examples of possible scenarios. Due to mentioned reasons, a new forest fire detection dataset was created by gathering forest fire images from Incident Dataset [38] and VIDI Dataset [39]. Incident Dataset is a large dataset that contains 446,684 positive class and 697,464 negative class images. This dataset consists of different incident and place categories as listed in Figure 2.1.

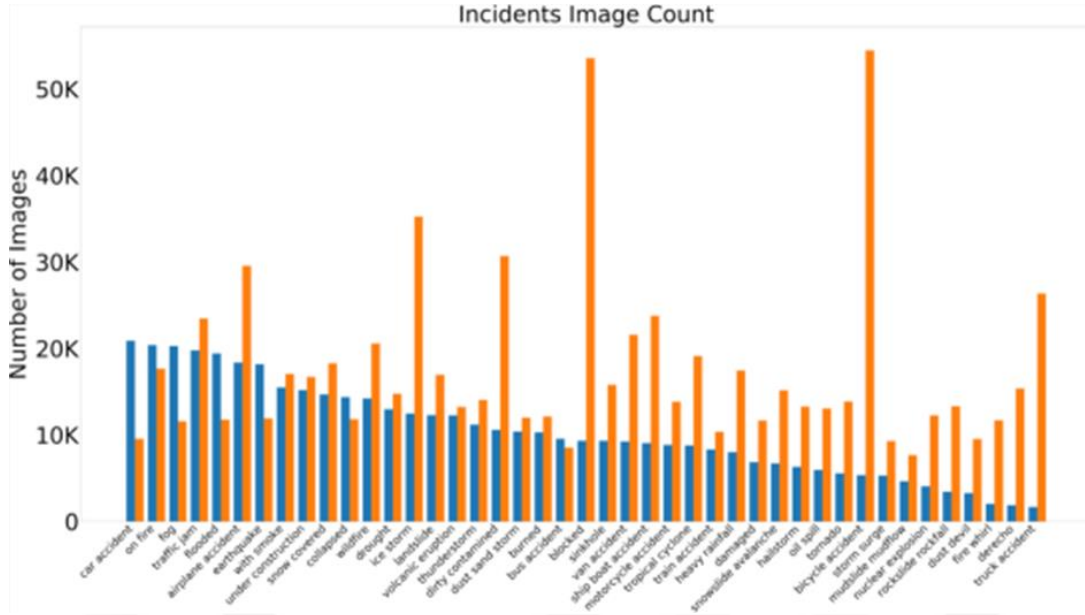


Figure 2.1: The incident dataset composition taken from [38].

The incident dataset was prepared by collecting images from different resources and labeled by humans. Some images have been containing more than one incident as in Figure 2.2. Class-negative images include both unrelated images and places without any incidents. Sample images from the dataset are given below.

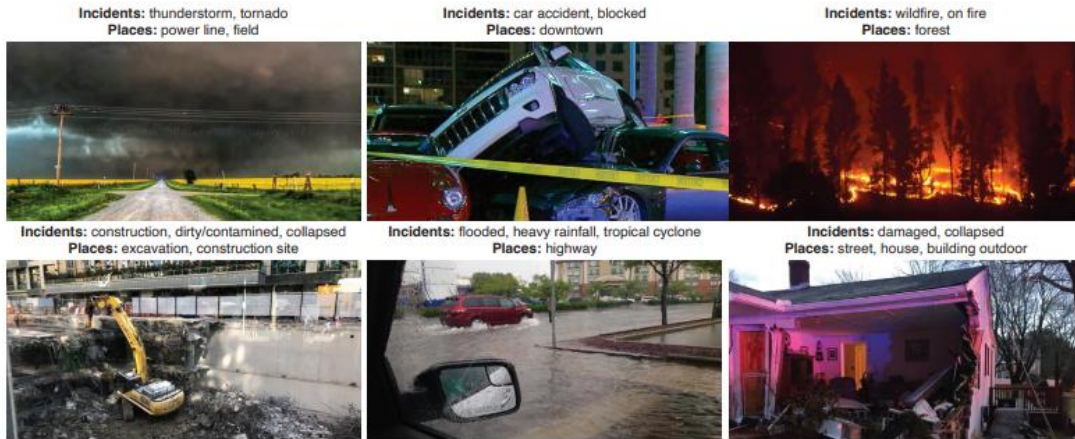


Figure 2.2: The incident dataset image samples taken from [38].

The incident and VIDI datasets are quite large and diverse with images containing many incidents and places. Since these datasets include various incidents, there is a need to extract forest fire images from them. Therefore, a dataset specific to forest fire detection was created using the images in Incident Dataset. In this section, forest fire detection is addressed as a binary classification problem. Thus, the positive class (fire) was created by wildfire-tagged class-positive images and the negative class (no-fire)

was created by forest-tagged class-negative images from the places category. Class-negative and class-positive image samples from the created new dataset are shown in Figure 2.3 and Figure 2.4.



Figure 2.3: Class-positive image samples (fire).



Figure 2.4: Class-negative image samples (no-fire).

The collected images contain 10,583 fire images and 11,849 no-fire images. Our dataset has more content compared to other datasets used in the literature. Images include close-up images of forest fires, as well as aerial images and smoke images during the fire. Thus, it was provided that models suitable for possible scenarios were created for real-time applications.

2.1.2 Image preprocessing

The images of the architectures we chose for the fire detection problem should be given to the model in a specific size, and in our work, the used networks need an input size of 224x224 or 299x299. All of the images in the dataset are of different sizes. Thus, the images have been resized to ensure regularity and make them suitable for the

requirements of the architecture. As another method, normalization was applied. Image normalization is a process that changes the range of intensity values from 0-225 to 0-1. Normalization provides faster convergence and increasing in computational efficiency. It can also help with preventing gradient explosion and vanishing problems. Different methods are used in the literature, such as linear scaling, min-max clipping, log scaling, and z-score. In this work, the Lp normalization methods used for normalizing images [40]. It means subtracting channel means (μ) values and dividing them into channel standard deviation (σ).

$$x_{new} = \frac{x - \mu}{\sigma} \quad (2.1)$$

2.2 Deep Learning Methods

2.2.1 Convolutional neural networks

Convolutional neural networks (CNN) are a type of neural network that is mostly used for computer vision tasks such as image classification and object detection. The benefits of using CNNs in computer vision tasks are reducing the number of parameters with weight sharing, faster implementation than other methods, and robust feature extractors [41]. Besides, one of the important advantages of CNNs is the feature extraction step. Feature extraction helps to determine important information, reduces redundant data in the images, and provides faster performance without any quality loss. The idea of feature extraction is that the model can learn itself from images what important features are. Hence, CNNs can detect high-level features without additional information, such as edges, shapes, motion etc. CNNs are consists of layers with different responsibilities as displayed in Figure 2.5, and we can mention them below:

-Convolutional Layer

-Pooling Layer

-Fully Connected Layer

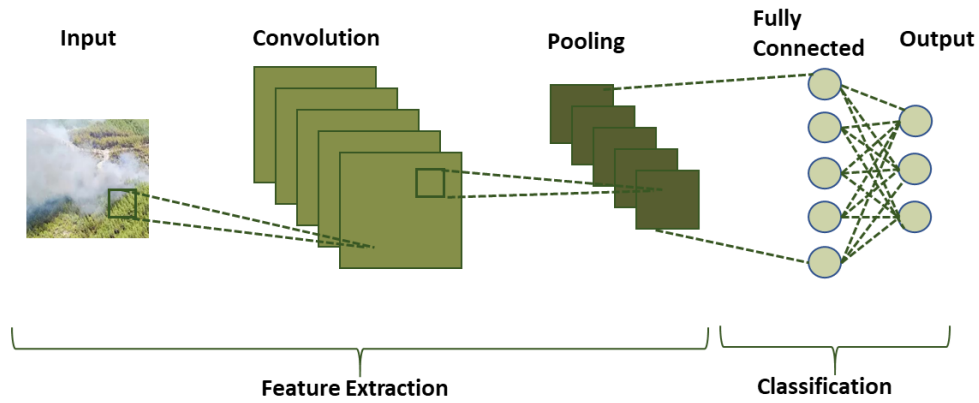


Figure 2.5: CNN Structure.

Convolutional layers are responsible for detecting patterns in the images by extracting low-level features and high-level features from each layer [42]. To apply convolution operation, there is a need to determine filters for creating a feature map. In a convolution operation kernel moves across the input image, taking the dot product of the two matrices and saving the values to a new matrix as the feature map of the original image. When kernels are loaded with the appropriate values and convolved with the input image produces an output which highlights to a specific feature of the image. Then, the weights of each layer are updated in every step in the end it is determined essential patterns in the image. We can summarize these operations by exemplifying the convolution operation in Figure 2.6.

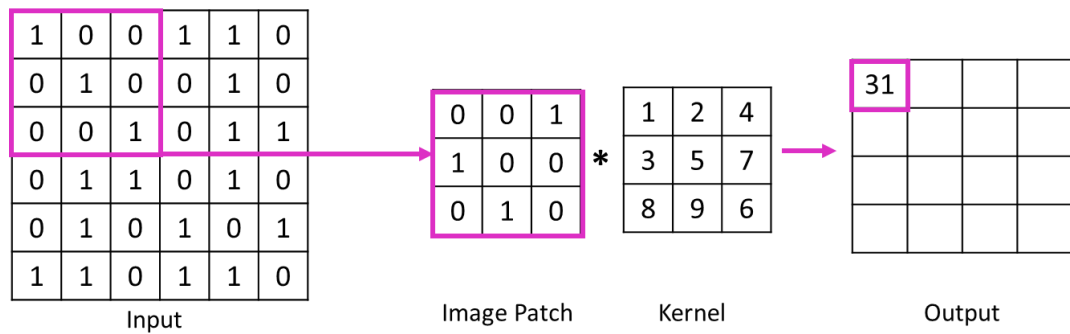


Figure 2.6: Convolution example.

The pooling layer provides down sampling feature maps to reduce the number of parameters and computational cost [43]. There are several types of pooling operations such as max pooling, min pooling, and average pooling. Figure 2.7 is showed max and average pooling calculations for better understanding. In many applications, max pooling is preferred rather than average pooling and min pooling. Max pooling takes

the largest element in the feature map while average pooling calculates the mean of the chosen areas and min pooling selects the smallest element. Benefits of the using max pooling, it gives more sharp images to determine important parts of the image and performs a noise compressor.

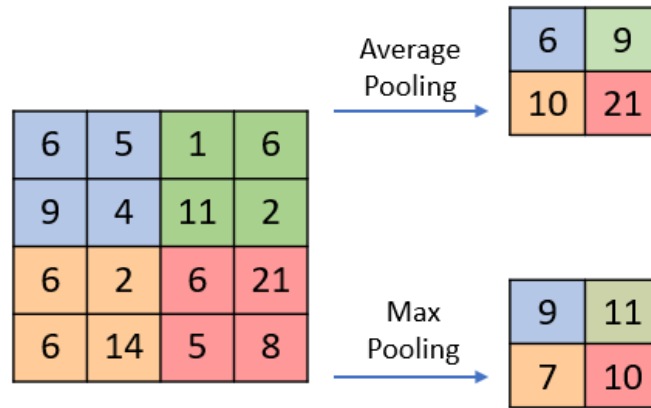


Figure 2.7: Pooling examples.

A fully connected (FC) layer in CNN is one that takes the result of the convolution or pooling layer and reaches a classification decision. The example structure of the FC layer is shown in Figure 2.8. These layers consist of weights and biases and are generally located before the output layer. The working principle of the fully connected layers is similar to artificial neural networks [44]. Information from previous layers is flattened into a column vector, and flattened output is joined backpropagation step. In the last step, the classification decision is made by calculating the class scores.

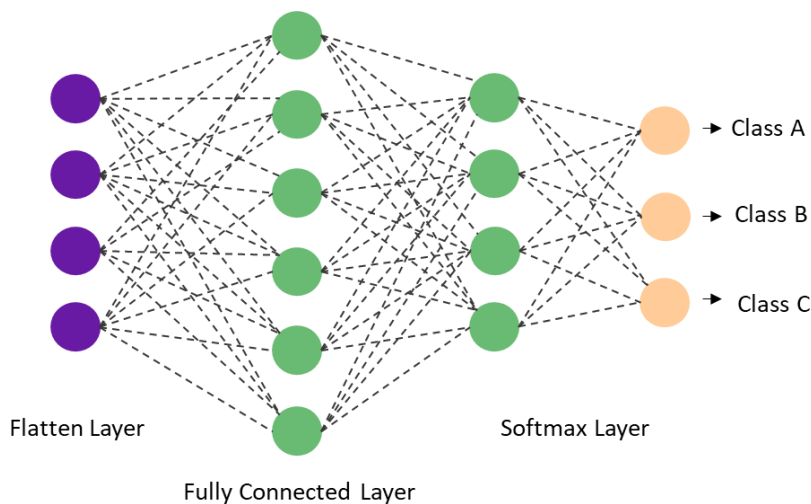


Figure 2.8: Example of FC layer architecture.

Another important concept in CNN architectures is activation functions. Various activation functions are used in artificial intelligence methods, and different activation functions can be used in each layer. Sigmoid, ReLU, tanh, Maxout, etc., are commonly used activation functions. Examples of these are given with their functions in Figure 2.9. The purpose of using these functions is to add non-linearity to the network so it would be capable of learning more complex patterns. Without non-linearity, the network is restricted to learning only linear functions.

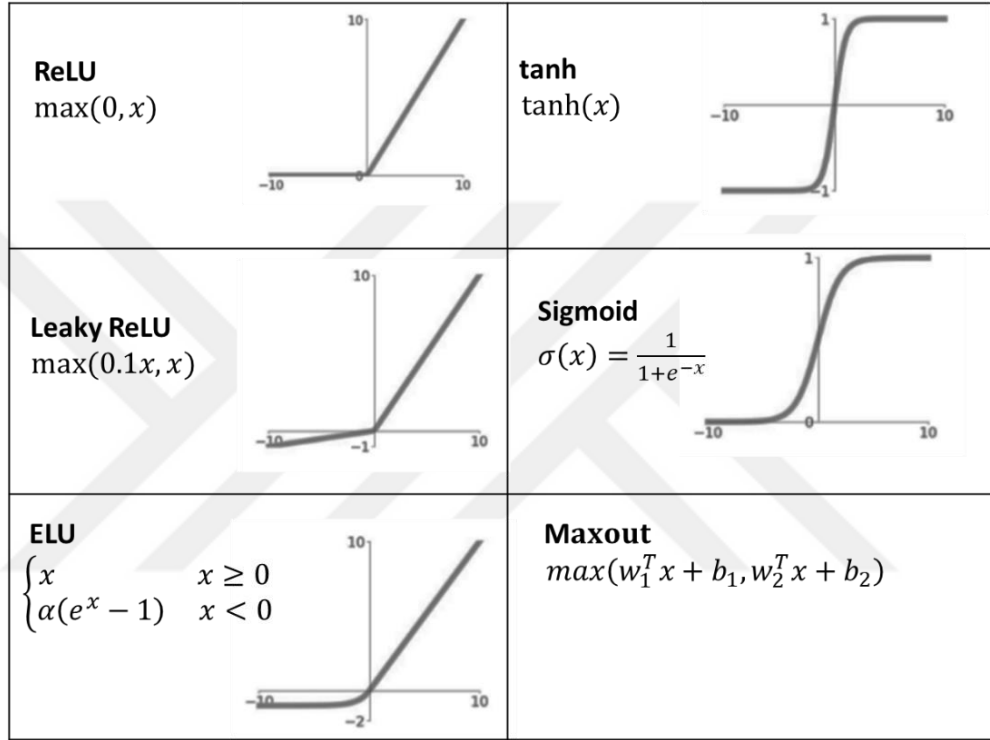


Figure 2.9: Activation function examples.

2.2.2 ResNet101

ResNet is a powerful deep convolutional neural network that won the ImageNet challenge in 2015. This network was introduced in a paper named "Deep Residual Learning for Image Recognition" by He et al [45]. Since creating deeper networks can provide better generalizations, ResNet focuses on building deep connections without losing any information. One of the biggest challenges in creating a deep network is the vanishing gradient problem. The weights of a neural network are updated using the backpropagation algorithm by making a small change to each weight to the loss of the model decreases. The weights of a neural network are updated using the backpropagation algorithm by making a small change to each weight to the loss of the model decreases. While updating weights, the gradient with respect to weights in the

initial layers of the network flows backward. Hence, when the network gets deeper, the gradient becomes smaller and may converge to zero; this is called a gradient vanishing. To overcome this problem, ResNet proposed residual blocks made from skip connections which can build alternate shortcut paths for the gradient to flow through. The mathematical representation of the residual block is given in below.

The output of a block of layer is represented as $H(x)$ and the residual block is express as Figure 2.10:

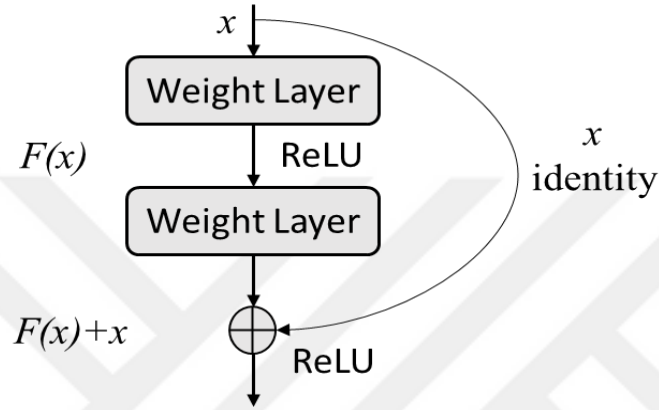


Figure 2.10: Residual block representation taken from [46].

where

$$\begin{aligned} H(x) &= F(x) + x \\ F(x) &= H(x) - x \end{aligned} \tag{2.2}$$

Then loss function becomes:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial H} \frac{\partial H}{\partial x} = \frac{\partial L}{\partial H} \left(\frac{\partial F}{\partial x} + 1 \right) = \frac{\partial L}{\partial H} \frac{\partial F}{\partial x} + \frac{\partial L}{\partial H} \tag{2.3}$$

The architecture of 34-layer ResNet is as follows Figure 2.11. ResNet has many variants with the same architecture design but different layer sizes, such as ResNet18, ResNet34, ResNet50, ResNet101, ResNet152, ResNet164 etc. In this work, ResNet101 was used for forest fire detection.

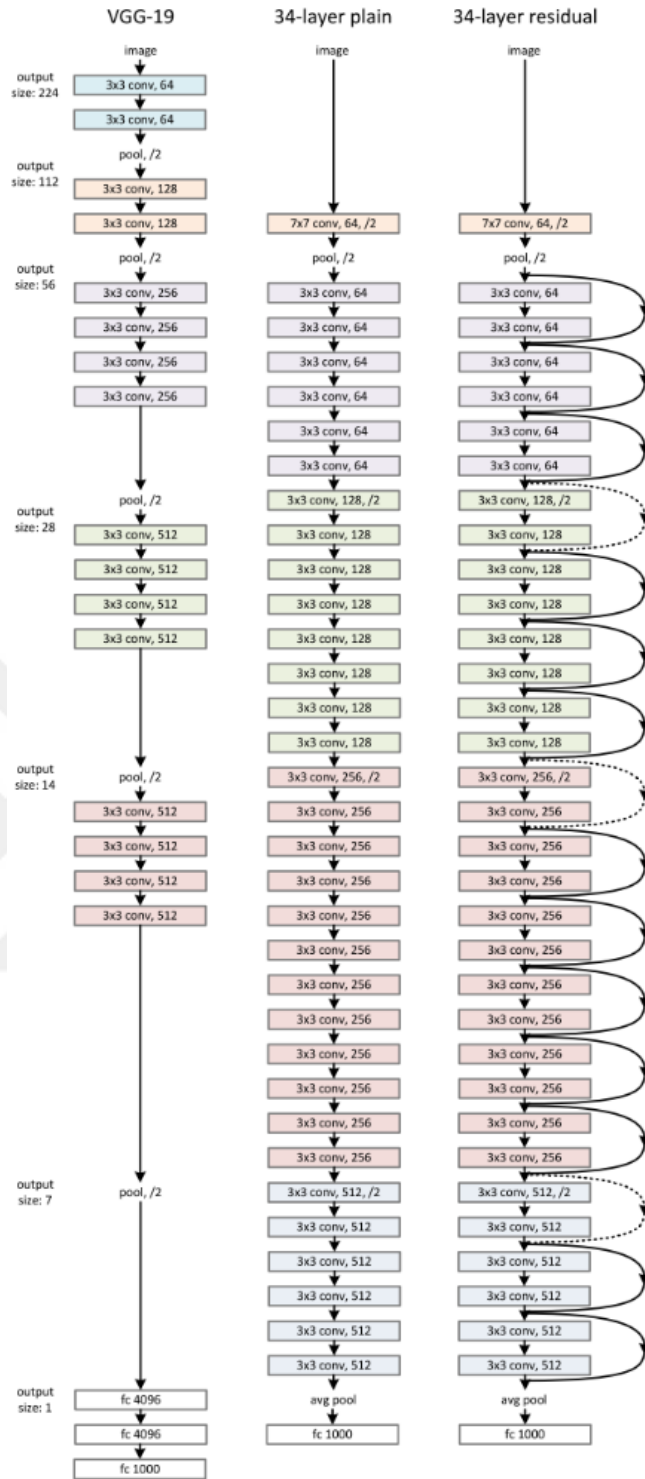


Figure 2.11: ResNet architecture taken from [45].

2.2.3 DenseNet

DenseNet is a deep convolutional neural network. The intuition behind this network allows information to be directly transmitted to later layers in a feed-forward fashion, and each layer receives collective knowledge from all the previous layers [47]. In this

way, compared to ResNet, which has a similar architecture, the number of parameters is reduced, and the network becomes more compact. As mentioned before, deep networks face the gradient vanishing problem. The initial layers in the network may lose some information, where the gradients become too small. DenseNet provides a solution to the vanishing gradient problem by using dense blocks. Dense blocks in DenseNet architecture allow all layers to have direct access to the original signal and gradients of the loss function [48]. Example dense block structure is shown as in the Figure 2.12. It provides mainly initial layers that can update more efficiently and allow better gradient flow. Another advantage of using DenseNet, it reduces parameter size compared to ResNet by reaching the same amount of model success. Decreasing the number of parameters increases the speed of the model. Since adding dense blocks increases the number of channels, and this may cause the model to become overly complex. Thus, transition layers are put in to balance the network. Using a 1×1 convolutional layer and an average pooling layer with a stride of two reduces the number of channels and model complexity.

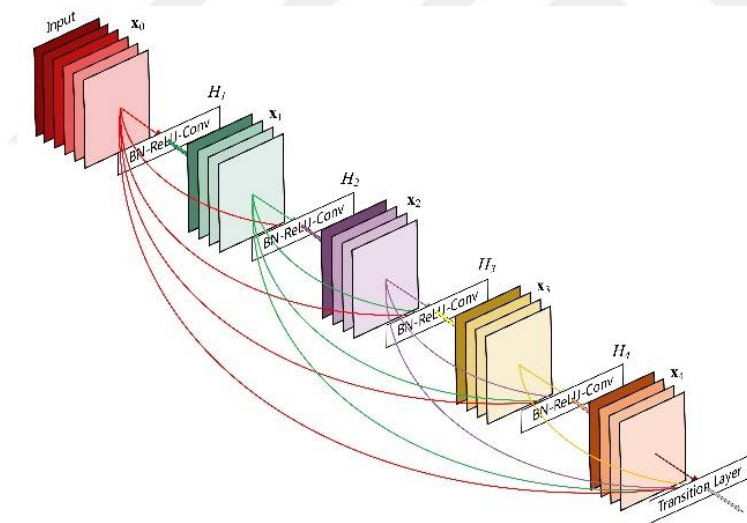


Figure 2.12: A 5-layer dense blok taken from [47].

2.2.4 Inceptionv3

Inception is a CNN that primarily focused on increasing the depth of the network considering avoiding overfitting. The Inception architecture focuses on parallel processing and concurrently extracting various feature maps [49]. It inspires by the network-in-network technique that uses the benefits of 1×1 convolution. This 1×1 filter is not used to learn spatial patterns. It provides a reduction in the dimensions and widened layers by applying parallel processing. The widening effect in Inception

modules is created by performing 1×1 , 3×3 , 5×5 filters and 3×3 max pooling operations in parallel in the convolution layers. This structure is called the "Naive Inception Module" as in Figure 2.13.

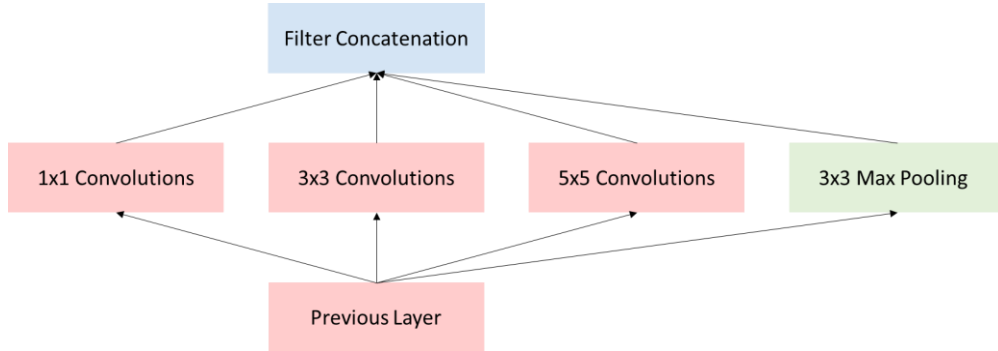


Figure 2.13: Naive inception module taken from [49].

Deepening and expanding networks is a computationally expensive process. Adding 1×1 filters to the module after the convolution operations reduces dimensions to avoid costly computation. Structure of this method given in Figure 2.14.

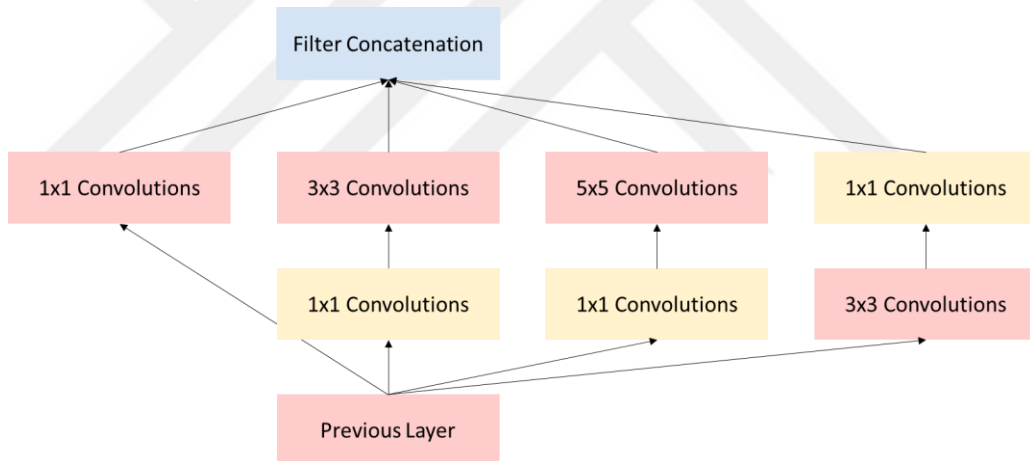


Figure 2.14: Dimension-reduced inception modules taken from [49].

Later versions of the Inception have improved the network to make it more computationally efficient. To achieve this, two 3×3 convolutional layers were used instead of 5×5 convolutional layers. Two 3×3 convolutional layers contribute equally to one 5×5 convolutional layer and can extract more features using fewer parameters. Inceptionv3 has smaller, asymmetric and factorized convolutions, uses an RMSprop optimizer, auxiliary classifiers and reduced grid size [50]. Auxiliary classifiers are small CNN located between layers to improve convergence. Since in early training auxiliary classifiers did not boost convergence Inceptionv3 uses them as regularizers. The architecture of Inceptionv3 is given Figure 2.15.

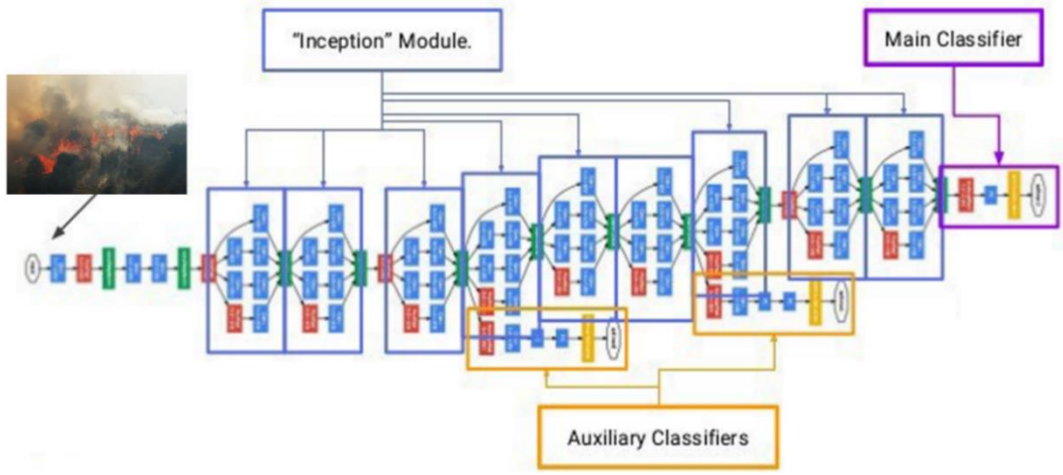


Figure 2.15: Inception architecture taken from [50].

2.2.5 RegNet-X-32GF

RegNet is a design space to create a highly flexible, accurate, and efficient network. It creates an environment to build an adjustable network by letting select the width and depth of the structures. Using this environment, networks can design by considering the efficiency of the network or classification performance. This structure is labeled as a design space instead of a network. Firstly, the authors have created different design spaces for all potential models called AnyNet [51]. It is determined what parameters are suitable and essential for performance by experimenting with different settings and design space built up with respected results. The AnyNet design space consists of three main parts stem, body, and head. The stem and head part are designed as simple as possible but the body part is more complex because it is responsible for accuracy and computation performance. Figure 2.16 is summarized AnyNet architecture.

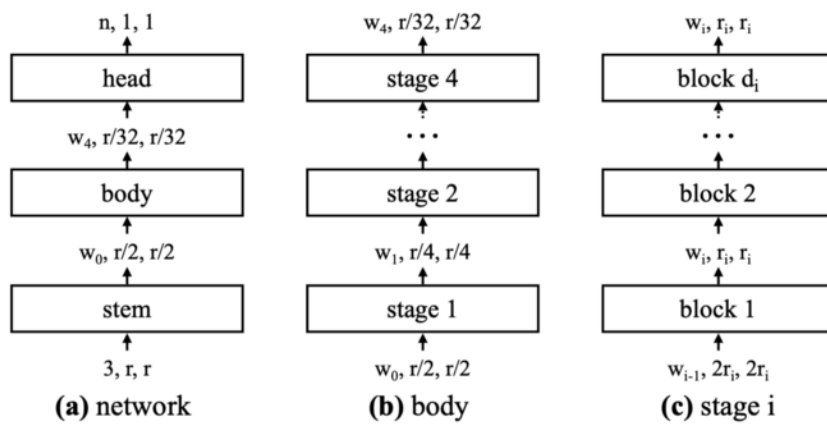


Figure 2.16: AnyNet architecture summary taken from [51].

AnyNet provides a lot of possibilities for creating new networks. Therefore, RegNet is presented with adding some restrictions to generalize this design space. The body part of RegNet has a standard residual bottleneck block with group convolution and includes composed of multiple blocks. Therefore, RegNet does not have fixed parameters, but it has quantized linear function adjusted with selected parameters. The block widths are calculated as follows:

$$\begin{aligned}
 u_j &= w_0 + w_a \cdot j \text{ for } 0 \leq j < d \\
 u_j &= w_0 \cdot w_m^{s_j} \text{ for } 0 \leq j < d \\
 w_j &= w_0 \cdot w_m^{\lfloor s_j \rfloor} \text{ for } 0 \leq j < d
 \end{aligned} \tag{2.4}$$

In the Equation 2.4 u_j is block width at stage j , w_0 the initial width and w_a the slope, w_m width parameter, j starts at 0 and ends at the depth of the network. RegNet is created by setting parameters such as depth, initial width, slope, width parameter, bottleneck and group number [52]. RegNet has two different versions as RegNet-X and RegNet-Y. All the given information and equations define RegNet-X. RegNet-Y is a modified version of RegNet-X with adding squeeze and extraction layers. The blocks of the RegNet-X with different s values are shown in Figure 2.17.

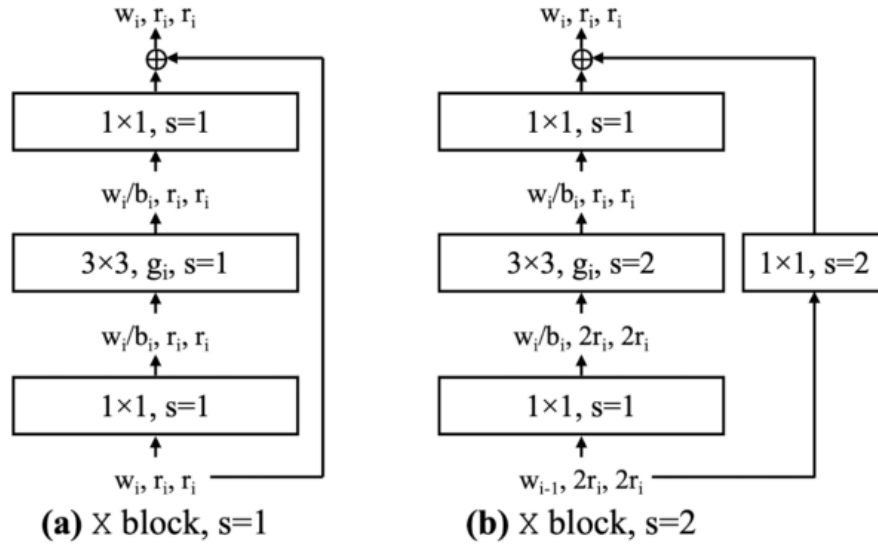


Figure 2.17: RegNet-X representation taken from [51].

In this thesis, the RegNet-X-based network, which optimal parameters set by authors as shown in Figure 2.18, is used to reach the highest accuracy and good classification results.

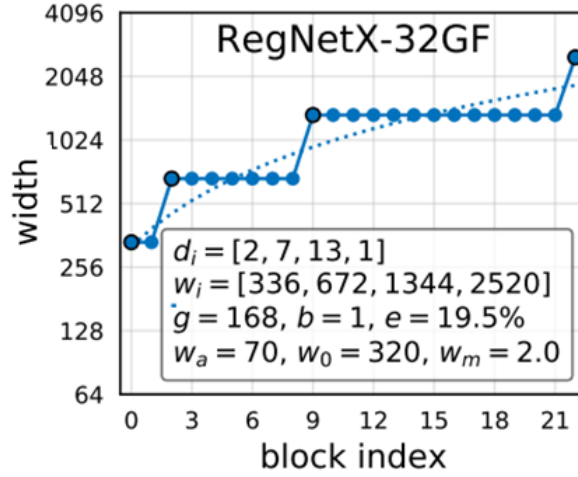


Figure 2.18: RegNet-X-32GF setup taken from [51].

2.2.6 Transfer Learning

The idea of transfer learning is to leverage the knowledge required by a model train with lots of data on another task. Transfer learning is commonly used in applications that require more computational power, such as natural language processing and computer vision tasks. Studies have shown that features learned from huge image sets, such as the ImageNet, are highly transferable to various recognition tasks [53]. By using pre-trained models, transfer learning can lead to increased accuracy compared to training from scratch. This is because the pre-trained models have already learned the fundamental features and representations of the data, which can be fine-tuned for the new problem. There are several ways we can transfer knowledge from one model to another. A common method in CNNs is to start with freezing the parameters of the pre-trained model to avoid losing all information from the model. Thus, the learned information is prepared to transfer to another task. Then classifier layers from the original model replace with the new trainable layers. Finally, the final model can be used for making predictions, or fine-tuning is applied by unfreezing the frozen layers to improve its performance. It gives the best results if data and tasks similar to the original model are trained on. Summary of the this methodology is given in Figure 2.19. Transfer learning is a functional method to save time. It is more advantageous to transfer the learned information for similar tasks instead of training the model from scratch. Creating a model with transfer learning enables successful generalization, especially in cases with small datasets.

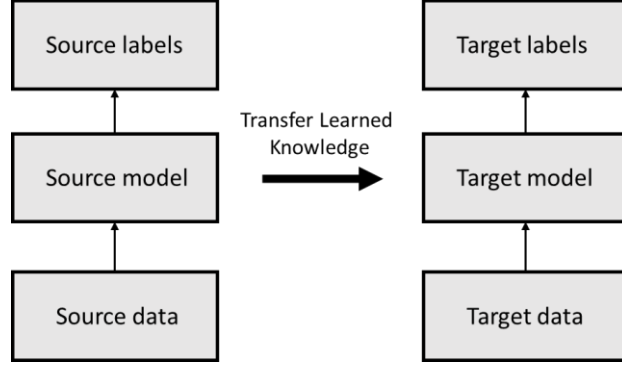


Figure 2.19: Transfer learning block diagram.

2.2.7 Evaluation Metrics

Evaluation metrics are used to measure the performance of models. Using multiple evaluation metrics gives more reliable results because using only one metric can be deceptive and biased. Therefore, this thesis used four evaluation metrics to test the success of the applied models: Accuracy, precision, recall, and F1 score. Firstly, these evaluation metrics are briefly explained.

- True Positive (TP): Model accurately predicts the positive class.
- True Negative (TN): Model accurately predicts the negative class.
- False Positive (FP): Model incorrectly predicts the positive class.
- False Negative (FN): Model incorrectly predicts the negative class.

Accuracy is the ratio of correct predictions to the total number and determines how well the model predicts.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.5)$$

Precision quantifies the percentage of positive class predictions that belong to the positive class and determines how well positives are predicted accurately.

$$Precision = \frac{TP}{TP + FP} \quad (2.6)$$

Recall is defined as the number of positives over the number of true positives and false negatives and determines how well actual positives are predicted correctly.

$$Recall = \frac{TP}{TP + FN} \quad (2.7)$$

The F1 Score is the harmonic mean of the precision and recall and it uses in imbalanced class distributions.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.8)$$

2.3 Optimization Techniques

2.3.1 Quantization

Quantization is a technique used for the optimization of neural networks to reduce memory usage and make faster implementation with losing minimum accuracy. In many applications, CNNs use 32-bit floats to represent parameters, but researches show that using high precision numbers is not always a requirement for high accuracy. High precision numbers are costly, slow to implement and increase memory usage. Considering these disadvantages, It will be more efficient to prefer lower numerical precision representation. Quantization aims to reduce the numerical precision of weights and activations in deep networks [54]. It can be applied with different methods to CNNs. The commonly used quantization mapping function is expressed in Equation 2.9:

$$Q(r) = round\left(\frac{r}{S}\right) - Z \quad (2.9)$$

Where Q is a quantization function, r real-valued input, S scaling factor, Z zero integer point. Round applies rounding operation to the nearest integer value. Input r can be activation or weight for the quantization function. This function is responsible for mapping real values to integer values. The output of the function can be uniformly spaced or non-uniform-spaced. The dequantization function in Equation 2.10 is used to recover real values.

$$\tilde{r} = S \cdot (Q(r) + Z) \quad (2.10)$$

The selection of the S factor is very critical and can be chosen symmetrically or asymmetrically [55]. $[\beta, a]$ is represented clipping range of the input which consists of real values. To understand the calculation of quantization for different schemes, affine and asymmetric quantization are explained below.

2.3.1.1 Affine/Asymmetric Quantization

The affine quantization scheme is implemented by detecting the minimum and maximum values. This method is highly efficient when quantizing non-negative activations.

$$f(x) = S \cdot x + Z \quad (2.11)$$

$$S = \frac{2^b - 1}{\beta - \alpha} \quad (2.12)$$

$$Z = -\text{round}(\alpha \cdot S) - 2^{b-1} \quad (2.13)$$

where $f(x)$ is represented as an affine transformation function (Equation 2.11), S is the scale factor (Equation 2.12), b is the bit-width of the signed integer and Z is zero-point (Equation 2.13). Then quantization operation is expressed by the following Equations 2.14 and 2.15 and the dequantization operation at Equation 2.16.

$$\text{clip}(x, l, u) = \begin{cases} l, & x < l \\ x, & l \leq x \leq u \\ u, & x > u \end{cases} \quad (2.14)$$

$$\begin{aligned} x_q &= \text{quantize}(x, b, S, Z) \\ &= \text{clip}(\text{round}(S \cdot x + Z), -2^{b-1}, 2^{b-1} - 1) \end{aligned} \quad (2.15)$$

$$\hat{x} = \text{dequantize}(x_q, S, Z) = \frac{1}{S}(x_q - Z) \quad (2.16)$$

In this method, the range expresses as $[\alpha, \beta]$. Asymmetric quantization applies fully utilization to the quantization range. It is more general and more flexible with extra offset. Besides, implementation of asymmetric quantization can be more costly than symmetric.

2.3.1.2 Symmetric Quantization

Symmetric quantization does not use centering the input range close to 0 as in affine quantization. Since this calculation includes values not in the range when non-negative activations are used, it may cause performance degradation in the results. Representation of symmetric range can be shown as $[-\beta, \beta]$. The calculations of the symmetric quantization parameters are as in the Equation 2.14, Equation 2.15 and Equation 2.16 given below.

$$S = \frac{2^{b-1} - 1}{\beta} \quad (2.14)$$

$$\begin{aligned} x_q &= \text{quantize}(x, b, S, Z) \\ &= \text{clip}(\text{round}(S \cdot x + Z), -2^{b-1} + 1, 2^{b-1} - 1) \end{aligned} \quad (2.15)$$

$$\hat{x} = \text{dequantize}(x_q, S) = \frac{1}{S} x_q \quad (2.16)$$

2.3.1.3 Per-tensor and Per-channel Quantization

Quantization can be applied in two different ways, per-tensor and per-channel as in Figure 2.20. In the per-tensor scheme, using the same clipping range for all tensors, separate quantization parameters are calculated for the weight tensors in the layers. In the per-channel scheme, they are calculated separately for each channel. Although two schemes can be useful depending on the problem, per-channel-based implementations provide better accuracy results. The reason for per-tensor may perform at a different level than per-channel is the high variance in the convolution weights due to the batchnorm folding.

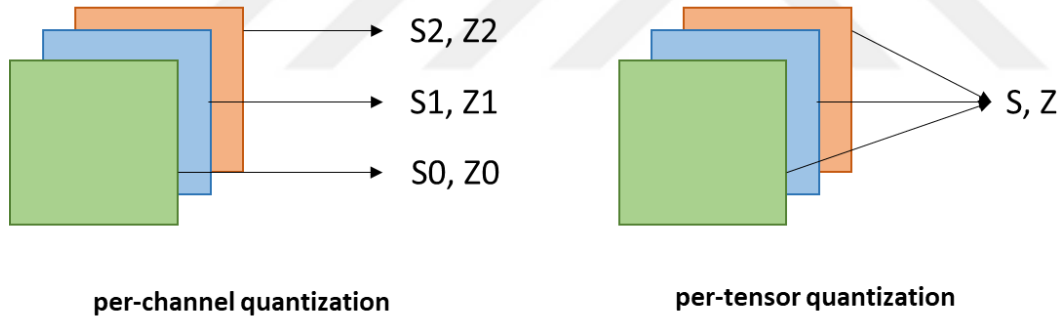


Figure 2.20: Per-tensor and per-channel quantization scheme representation.

Determining the way of quantization is another critical differentiator after the clipping range. It can be performed statically or dynamically. Static quantization, quantizes the activation and weights of the model by fusing activations into preceding layers [54]. This method requires an additional calibration step to calculate optimal quantization parameters. Dynamic quantization applies quantization during runtime. This method makes calculations step by step with real-time computation. It is generally used in LSTM and transformer-type models. Both methods have advantages and disadvantages but the key is to choose the right method understand to the behavior of the model. Although dynamic quantization has the potential of reaching a higher

accuracy rate, it is seen that the static quantization method is frequently used because dynamic quantization is quite expensive to implement. Since we are working with CNNs, static quantization is chosen for model optimization. Another essential step of quantization is deciding whether to apply quantization after or during the training phase. The form of quantization applied during training is called quantization-aware training (QAT). It needs some additions for the training structure as fake quantization and quantize/dequantize nodes as shown in Figure 2.21 [56]. All weights and activations are fake quantized with float values changed to integer numbers without dramatically accuracy lost. Since the model is quantized from the training stage, the most minor loss in accuracy is achieved with this method, but it desires considerable computational power in terms of application.

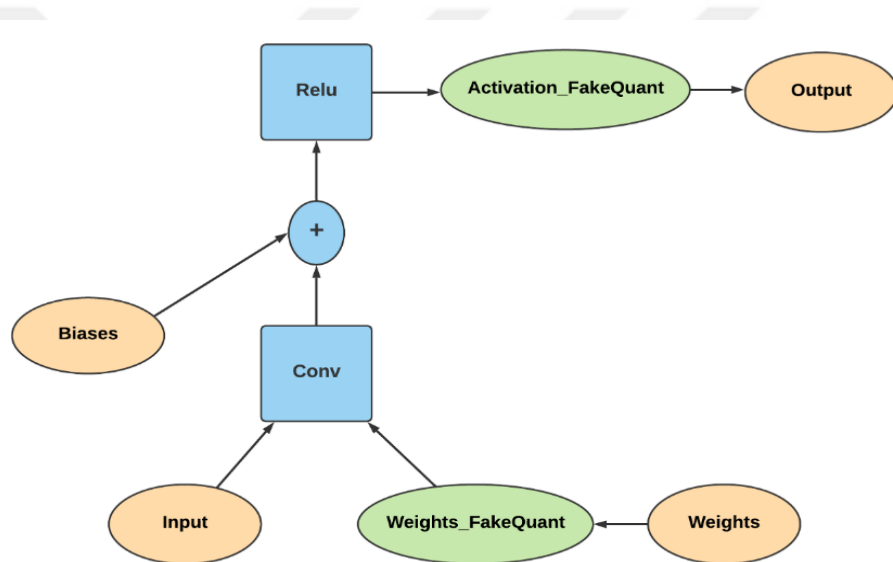


Figure 2.21: QAT configuration graph taken from [57].

Post-training quantization (PTQ) is an effective quantization method that apply quantization to a pre-trained float32 network without additional training steps [57]. The important point in PTQ is to determine quantization ranges for each quantizer because the range of quantizers affects quantization error. Besides, PTQ needs calibration of the data to avoid a drastic reduction in accuracy. Calibration is a method of determining the input clipping range. For this step, there are different approaches such as entropy minimization, mean-square-error minimization, determining min-max values for α and β , or percentiles of the input range. QAT and PTQ steps are presented in Figure 2.22 for making a better comparison between them.

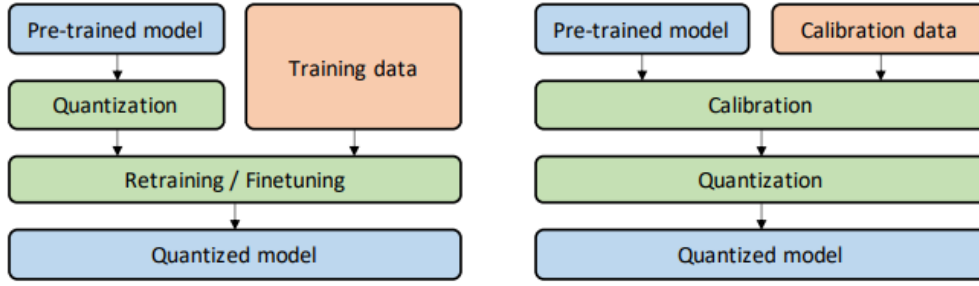


Figure 2.22: PTQ pipeline on the right and QAT pipeline on the left taken from [57].

In practice, before applying the post-quantization to the model, it needs to be preprocessed to avoid dramatically dropping accuracy. Besides to calibration step, layer fusion operation can be applied. Layer fusion applies as some layers are merged by grouping, and quantization is applied to the groups simultaneously instead of being applied to each layer independently. Fusion is an optional step but helps improve the speed of the model, accuracy, and memory access. QAT method provides the lowest accuracy drop and robustness but is costly, while PTQ can reach the same level of accuracy with some adjustments and is easier to use. Therefore, we used PTQ in our models.

2.3.2 Pruning

The idea of pruning is to eliminate redundant parts without affecting the performance of the network. Many deep learning applications use deep networks with many parameters, which paves the way for the highly accurate prediction of models. At the same time, these networks are costly and not easily applicable for low-power devices. Therefore, reducing costs in the network can increase the usage of state-of-the-art networks in real-time applications. The lottery ticket hypothesis helps to understand why pruning is a helpful method. The lottery ticket hypothesis argues that when working with large networks, there are lucky combinations in the initialization step [58]. Lucky initialization can provide high accuracy predictions that can be made by removing even more than 90% of the nodes in the network. In addition, pruning can reduce inference and training time, heat dissipation on hardware, and power saving. These benefits make deployed hardware applications more efficient. Different approaches can be used to perform the pruning by adjusting sparsity structure, scoring, scheduling, and fine-tuning [59]. Pruning can be fundamentally implemented as unstructured or structured as shown in Figure 2.23.

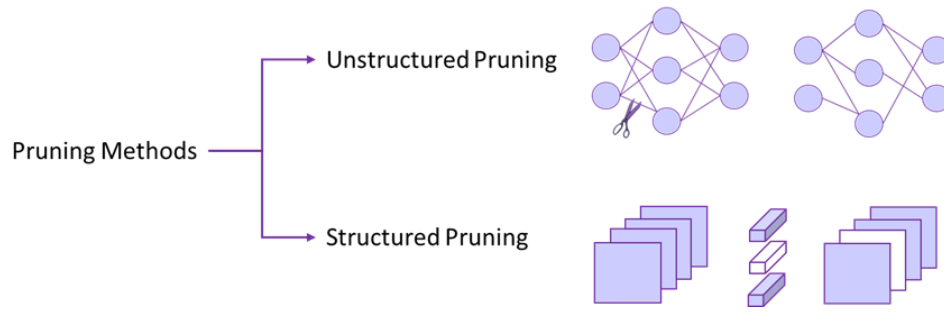


Figure 2.23: Pruning methods summary.

Unstructured pruning sets unnecessary connections in a network to zero, equivalent to switching off some of these connections and hence ignoring their effect. It can be applied with only one pass or recursively, depending on the tolerance in the trade-off curve. Visualization of unstructured pruning can be shown in Figure 2.24. A drawback of unstructured pruning is hardware and some frameworks do not support sparse tensor multiplications. On the other hand, the idea of structured pruning is to remove the entire units or channels. This results in changing the input as well as the output shape of the weight matrices. Although the concept of eliminating nodes seems similar to the dropout method, it has differences. While dropout nodes are restored during inference, pruning removes entire filters, channels, or neurons by setting the weight matrix to zero.

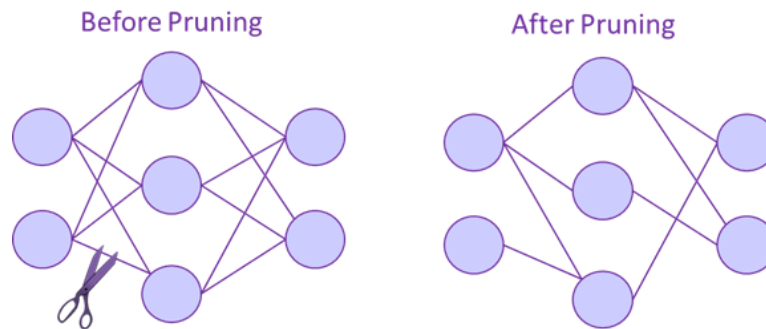


Figure 2.24: Unstructured pruning representation.

The essential manner in pruning is how to decide which node to keep and which one to prune. There are different approaches to this problem, and the most popular approaches are weight magnitude criterion, gradient magnitude pruning, random selection, and global/local pruning. Gradient magnitude pruning is based on the Taylor decomposition method to prune nodes without damaging the network. The second derivative of the objective function of the network is computed to calculate the saliency, and low saliency parameters are removed after each step. Magnitude-based

pruning scheme calculates the minimum absolute value of weights. This method removes all weights whose magnitudes are below a specific threshold, and the remaining weights are readjusted in the next round of training. The major drawback of the approaches mentioned is that they require the model to be retrained, resulting in performance overhead. Global pruning applies chosen criteria to all parameters while local pruning prefers to use criteria individually. The final concept of the pruning criteria is whether the chosen criterion is applied to all parameters in the network or individually for each layer. Although global pruning can give better results, may occur layer collapse. To prevent collapsing in some layers local pruning should be applied to the network. Combining two approaches can give the best results for hardware implementations. The structure of the general pruning implementation is followed by training, pruning, and fine-tuning as in Figure 2.25. The determination of the mentioned criteria depends on the application area. Every problem has its unique solutions. Pruning can be performed as one-shot pruning or iterative pruning. Iterative pruning requires re-training, which adds extra effort to implementation. In addition, pruning has some disadvantages such as poor quantification of the impact of compression beyond overall accuracy, potentially little speed-up gains, hard to select compression method without doing some experiments. Therefore, pruning is a promising technique that reduces the inference time and complexity of the network for limited sources of hardware applications.

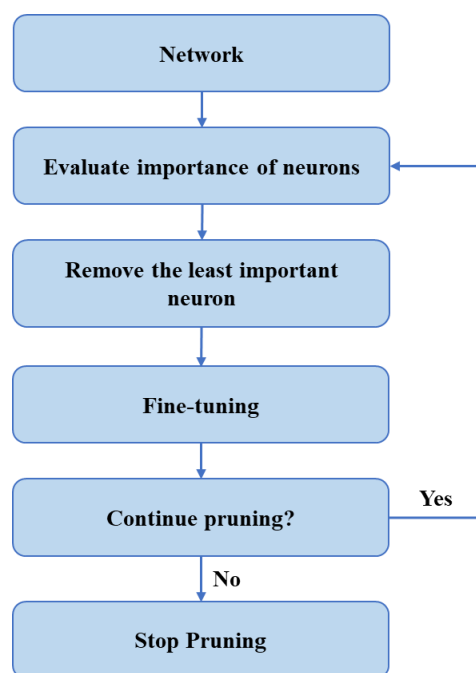


Figure 2.235: Pruning implementation pipeline.

2.4 Results and Discussion

Dataset has been prepared by extracting images from the Incident and the VID1 datasets of forest fires. These images contain flame and smoke images from different angles and places to train the model for all possible situations during a forest fire. The collected images were preprocessed to make them suitable for the models used. Models should be fed with fixed-sized inputs such as ResNet101, RegNet-X-32GF, DenseNet-169 224 x 224 pixels, and Inceptionv3 299 x 299 pixels. Therefore, the images have been reshaped during the training phase according to these sizes. In the next step, the reshaped images were normalized with the Lp normalization technique to avoid gradient explosion and vanishing problems.

In this thesis, studies have been focused on detecting forest fires fast, reliable, efficient and the most accurate way. In recent years the use of low-power devices with resource limitations has increased for the early detection of forest fires. Therefore, this thesis aims to make complex architectures that require computational power more suitable for low-power devices and prepare them for real-time applications. In this direction, ResNet101, RegNet-X-32GF, Inceptionv3, and DenseNet-169 networks with different working principles were chosen to detect forest fires. These networks were prepared for forest fire detection by using transfer learning to get the benefits of these networks' pre-trained versions. To summarize these architectures, ResNet has residual connections which provide a solution to the gradient vanishing problem. Since there are various versions in terms of size, we used the ResNet101 version to reach the most successful accuracy. Then, the DenseNet architecture consisting of dense blocks was implemented. This network needs less memory space than other architectures and still achieves high accuracy. Another approach, Inceptionv3, which has dimension-reduced inception modules to reduce the computational requirements of the model, was employed. Finally, unlike the other networks, RegNet-X-32GF was created from the design space by selecting the optimal parameters for using forest fire detection. In the first step, the networks are trained in plain form without adding any optimization method. One fully connected layer was replaced with the last layer of the networks with sigmoid activation. Then, we divided our dataset into 70% train, 15% test, and 15% validation sets. Using the validation set is an effective method for creating an unbiased model. The validation set is used to measure the performance of the model during training. After each epoch, we create a checkpoint for the validation loss and

observe training progress. One of the important points during the training phase is selecting the loss function and the optimizer. Binary cross entropy (BCE) loss is chosen for loss calculations. It takes each class independently which means the output of the CNN does not affect by other component values. Then, the Adam optimizer was used as the optimizer which uses the adaptive moment estimation technique. The training was implemented with 70 batches and 40 epochs on the NVIDIA Titan RTX GPU. Results of the training and validation are given in Figure 2.26 with sharing their loss graphs. Then finally, the hyperparameters and configurations of the model are determined. Then, their performances were evaluated on the test set.

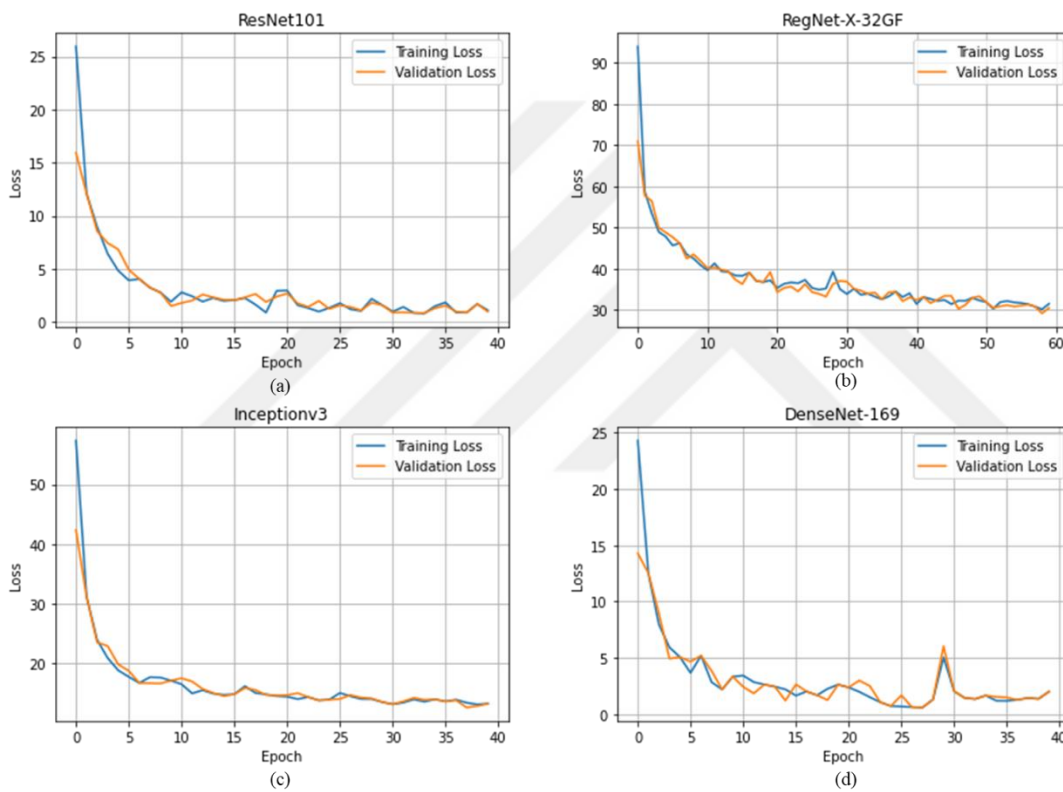


Figure 2.246: Training and validation loss graphs for ResNet101 (a), RegNet-X-32GF (b), Inceptionv3 (c), DenseNet-169 (d).

When the training and validation graphics are examined, it was observed that a decrease in the loss at each epoch shows the model fits the train test. The reduction in the training loss shows that the model makes wrong predictions less in each epoch, and after a certain number of epochs, the model reaches its optimal form. After the model with the most accurate estimation parameters was selected, the performance of the model in the unseen images was evaluated using the test set. The training and test performance results are reported in Table 2.1.

Table 2.1: Performance evaluation results of vanilla models.

	Accuracy		Precision		Recall		F1-Score	
	<i>Training</i>	<i>Testing</i>	<i>Training</i>	<i>Testing</i>	<i>Training</i>	<i>Testing</i>	<i>Training</i>	<i>Testing</i>
ResNet-101	0.992	0.980	0.997	0.981	0.999	0.983	0.998	0.982
RegNet-X-32GF	0.966	0.957	0.942	0.941	0.992	0.983	0.967	0.962
Inception-v3	0.998	0.973	0.998	0.978	0.997	0.978	0.998	0.976
DenseNet-169	0.992	0.976	0.994	0.969	0.996	0.984	0.993	0.977

According to Table 2.1, the models' prediction results are quite successful. Since no overfitting problem was encountered during the training, adding an additional regularization technique was not implemented. With their ResNet residual block and DenseNet dense block, and Inceptionv3 factorized convolutions structures, they prevented gradient vanishing and diminishing problems and reached high accuracy scores.

In the next step, quantization and pruning techniques with different configurations were applied to gain memory space, fast implementation and improve inference time. Experiments were made to find the most suitable pruning and quantization schemes for forest fire detection applications. Firstly, pruning was used to find redundant nodes and eliminate them in the network. While some techniques for applying pruning remove determined redundant nodes from the architecture, in this work we set these nodes to zero to protect the structure of the architecture. This way, we aimed to avoid the accuracy drop. In addition, applying quantization after pruning needs to be the whole architecture without any connection loss. Therefore, pruning is implemented in networks by setting redundant nodes to zero. L1 unstructured, random unstructured, L1 structured, and random structured pruning methods were applied by increasing the pruning ratios from 0.1 to 0.9. In Figure 2.27 present to the effect of change in compression ratio on accuracy. The graphs demonstrate the test results obtained after fine-tuning the pruned model. In this study, the models were retrained and finalized after pruning to get more robust results. In order to find the most suitable compression ratio, pruning was applied to the models with different ratio values. As the compression ratio increased, the models' accuracy leaning to decrease. In addition, it is observed that random selection methods lose more accuracy as the pruning ratios

increase. This is an expected result since randomly selecting the node may cause the necessary nodes to be set to zero in the network.

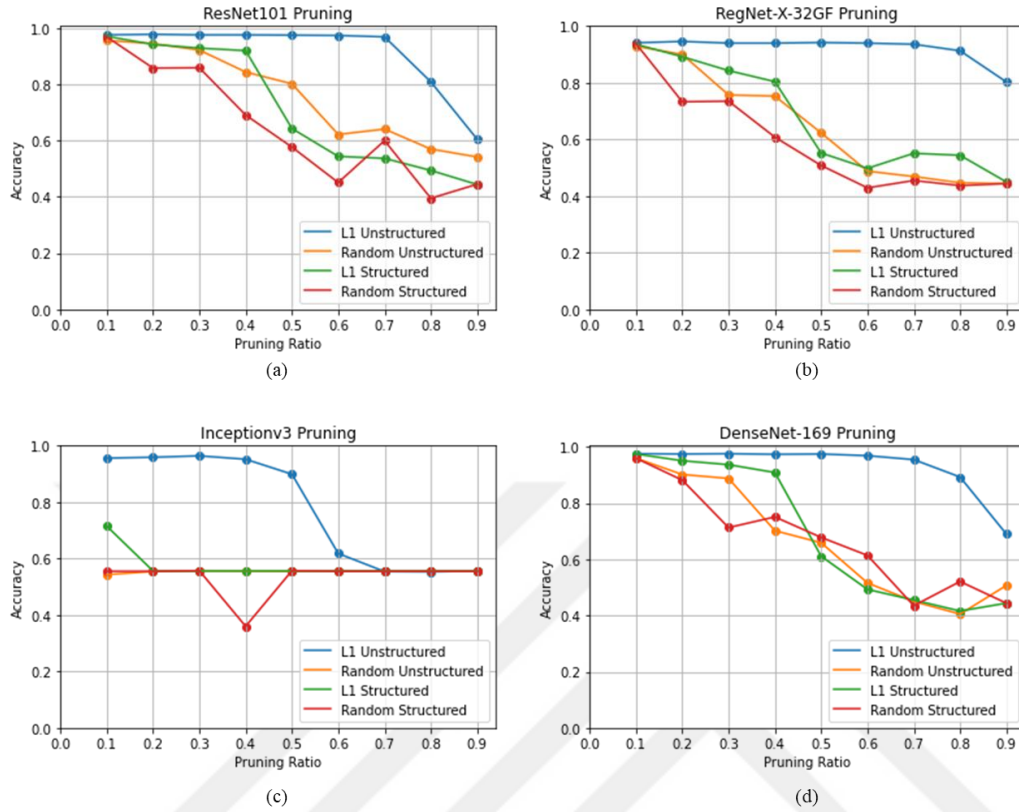


Figure 2.257: Pruning ratio vs accuracy graphs for ResNet101 (a), RegNet-X-32GF (b), Inceptionv3 (c), DenseNet-169 (d).

Hence, the L1 norm-based methods are the more reliable for node selection. When we examine the overall schemes of pruning, overall results show L1 unstructured method is the most resilient pruning technique for architectures. It was able to apply pruning at high compression ratios with minimum drops in accuracy. Specifically, when the results of the L1 unstructured method were compared, all models showed a linear decline in the high compression values. While this decrease was most in ResNet101, RegNet-X-32GF and DenseNet-169 showed quite comparable decreases. While ResNet101 has the most notable change in the accuracy, RegNet-X-32GF and DenseNet-169 showed quite comparable decreases. On the other hand, Inceptionv3 has more accuracy drop and falls behind the other networks in pruning compatibility. Then to examine the memory improvements, model sizes were calculated after pruning. As seen in the graphs in Figure 2.28, when the compression ratio increases, the space requirement of the models becomes less than their original versions. All the

pruning techniques show comparable results. However, random unstructured has slightly better improvements in memory size reduction compared to other techniques.

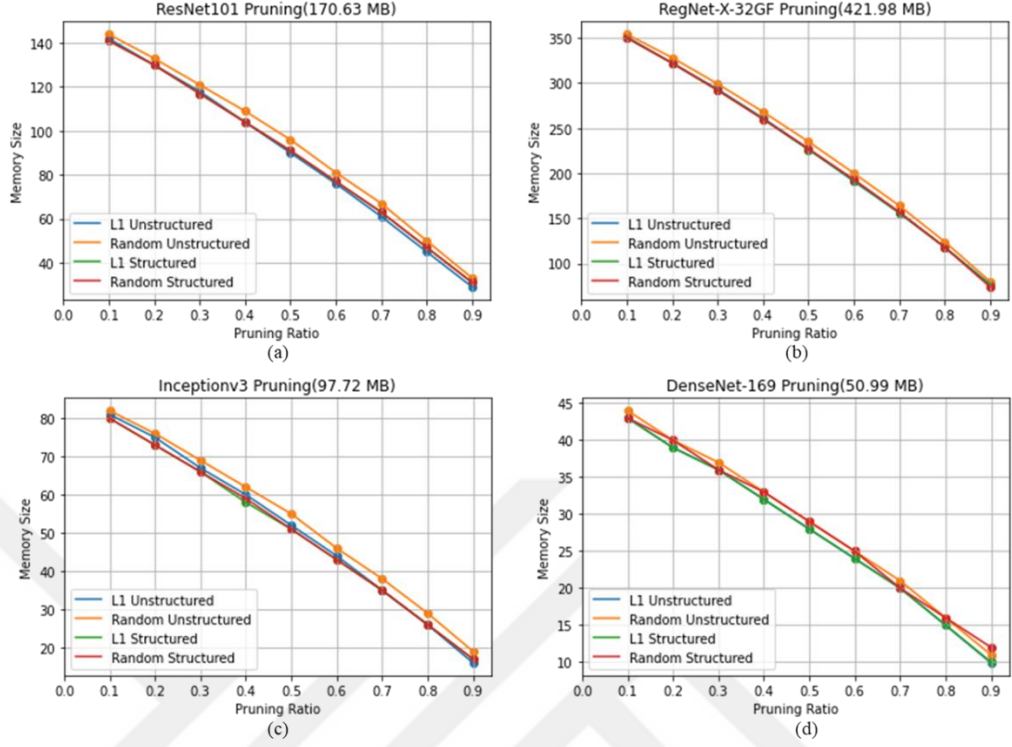


Figure 2.268: Pruning ratio vs memory size graphs for ResNet101 (a), RegNet-X-32GF (b), Inceptionv3 (c), DenseNet-169 (d).

Quantization was applied with different configurations as symmetric per-tensor, affine per-tensor, symmetric per-channel, and affine per-channel schemes. These configurations were used to find suitable quantization techniques for each architecture. All schemes are implemented by using post-static quantization. In order to prevent a drop in accuracy, the trained models were preprocessed before they were quantized. Thus, layer fusion is applied to the model by merging some layers before quantizing. Fusing layers are determined considering the architecture of the model, and it must be applied fixed sequences. Used sequences in ResNet101, Inceptionv3, DenseNet-169, and RegNet-X-32GF are given below.

ResNet101 → [Convolution, Batch Normalization, ReLU] and [Convolution, ReLU]

Inceptionv3 → [Convolution, Batch Normalization]

RegNet-X-32GF → [Convolution, Batch Normalization, ReLU] and [Convolution, Batch Normalization]

DenseNet-169 → [Convolution, Batch Normalization, ReLU] and [Batch Normalization, ReLU]

Then, calibration was applied to the models. For calibration, the min-max observer was used by collecting statistics on the train data. After calibration, the model has ready for quantization. At the end of all the quantization steps, the model was converted from float32 to int8 format. Performances of quantized models were evaluated by using the same test set used in vanilla models. Results are reported in Table 2.2, Table 2.3, Table 2.4 and Table 2.5 as different quantization schemes for all models.

Table 2.2: Performance evaluation of per-tensor affine quantization.

Quantization Schemes	Methods	Metrics			
		<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
Per-tensor Affine	<i>ResNet101</i>	0.951	0.927	0.987	0.957
	<i>RegNet-X-32GF</i>	0.945	0.984	0.915	0.948
	<i>Inceptionv3</i>	0.943	0.949	0.948	0.949
	<i>DenseNet-169</i>	0.924	0.882	0.997	0.936

Table 2.3: Performance evaluation of per-tensor symmetric quantization.

Quantization Schemes	Methods	Metrics			
		<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
Per-tensor Symmetric	<i>ResNet101</i>	0.624	0.596	0.999	0.746
	<i>RegNet-X-32GF</i>	0.939	0.959	0.930	0.945
	<i>Inceptionv3</i>	0.955	0.963	0.956	0.959
	<i>DenseNet-169</i>	0.804	0.740	0.998	0.850

Table 2.4: Performance evaluation of per-channel affine quantization

Quantization Schemes	Methods	Metrics			
		<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
Per-channel Affine	<i>ResNet101</i>	0.938	0.904	0.993	0.947
	<i>RegNet-X-32GF</i>	0.906	0.987	0.842	0.909
	<i>Inceptionv3</i>	0.922	0.926	0.934	0.930
	<i>DenseNet-169</i>	0.963	0.946	0.990	0.968

Table 2.5: Performance evaluation of per-channel symmetric quantization

Quantization Schemes	Methods	Metrics			
		<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1 Score</i>
Per-channel Symmetric	<i>ResNet101</i>	0.602	0.581	1.0	0.735
	<i>RegNet-X-32GF</i>	0.905	0.962	0.863	0.910
	<i>Inceptionv3</i>	0.926	0.908	0.965	0.936
	<i>DenseNet-169</i>	0.756	0.695	0.999	0.819

According to the quantization results, optimized ResNet101 and DenseNet-169 achieved the highest accuracies. When we examined the results in more detail, it was observed the per-tensor affine quantization has the least effect on the accuracy of each architecture from all schemes. On the other hand, per-tensor symmetric caused performance loss for ResNet101 and DenseNet-169 architectures, while Inceptionv3 and RegNet-X-32GF were able to predict with less accuracy drop. However, the per-channel affine shows the best quantization results for ResNet101 and DenseNet-169, and it also provided minimal accuracy drops for other architectures. As another scheme, the per-channel symmetric quantization caused more accuracy drops for ResNet101 and DenseNet-169, while Inceptionv3 and RegNet-X-32GF were less affected. Thus, symmetric-based quantization techniques are ineffective configurations for the ResNet101 and DenseNet-169 architectures. Although RegNet-X-32GF and Inceptionv3 remained above 90% accuracy in all quantization methods, when proper quantization techniques were selected, ResNet101 and DenseNet-169 reached the highest accuracy rates as if almost no optimization was applied. In summary, RegNet-X-32GF and Inceptionv3 would be more reliable choices if quantization configurations were limited. However, if customization is not limited, it is more advantageous to use ResNet101 and DenseNet-169 with proper schemes. When examined in general, affine quantization can show better results for all architectures because it makes mapping calculations with considering data. Besides per-channel approach may provide quantization without damaging the performance of the model by calculating scale and zero factors individually for channels. The comparison of the original models and optimized models is presented in Table 2.6 for better understanding.

Table 2.6: Original and quantized models comparison.

Methods	Accuracy				
	No Quantization	Per-tensor Affine	Per-tensor Symmetric	Per-channel Affine	Per-channel Symmetric
<i>ResNet101</i>	0.980	0.951	0.624	0.928	0.602
<i>RegNet-X-32GF</i>	0.957	0.945	0.939	0.906	0.905
<i>Inceptionv3</i>	0.973	0.943	0.955	0.922	0.926
<i>DenseNet-169</i>	0.976	0.924	0.804	0.963	0.756

The main point of this study is to improve the speed and memory space of the architectures with a minimum amount of loss in accuracy. With this motivation, it is essential to observe and interpret the effects of these quantization techniques on memory and inference time. Firstly, non-quantize models' memory space and speed performances are calculated. Then, per-tensor symmetric, per-tensor affine, per-channel symmetric, and per-channel affine quantization applied models are used for the same calculation. As expected, it was observed that all models got approximately four times smaller since the transition from 32-bit representation to 8-bit representation. Besides, although the improvement rates in the speed are comparable, it has been observed that they differ according to the structure of the model and the quantization method. The effects of these schemes on memory and inference time are given in the Table 2.7, Table 2.8, Table 2.9, Table 2.10, and Table 2.11.

Table 2.7: Size and inference time of non-quantized models.

Quantization Schemes	Methods	Metrics	
		Size (MB)	Inference Time (ms)
No Quantization	<i>ResNet101</i>	170.63	71.2
	<i>RegNet-X-32GF</i>	421.98	150
	<i>Inceptionv3</i>	97.72	28.5
	<i>DenseNet-169</i>	50.99	49

Table 2.8: Size and inference time results of per-tensor affine quantization.

Quantization Schemes	Methods	Metrics			
		Size (MB)	Size Difference(%)	Inference Time(ms)	Inference Difference (%)
Per-tensor Affine	<i>ResNet101</i>	42.76	74.93	29.6	58.42
	<i>RegNet-X-32GF</i>	105.58	74.97	63.5	57.66
	<i>Inceptionv3</i>	21.95	77.53	13.3	53.33
	<i>DenseNet-169</i>	13.82	72.89	21.1	56.93

Table 2. 9: Size and inference time results of per-tensor symmetric quantization.

Quantization Schemes	Methods	Metrics			
		Size	Size Difference(%)	Inference Time	Inference Difference (%)
Per-tensor Symmetric	<i>ResNet101</i>	42.76	74.93	28.2	60.39
	<i>RegNet-X-32GF</i>	105.58	74.97	63	57.99
	<i>Inceptionv3</i>	21.95	77.53	7.8	72.63
	<i>DenseNet-169</i>	13.82	72.89	20.5	58.16

When we examined per-tensor-based methods, the models were compressed at the same rate, and inference time changed with different ratios. Inception was the fastest and the greatest improved architecture in all of them. Other methods also gave quite successful improvements. However, in the decision point for selecting the most suitable configuration, there is a need to consider accuracy scores with these results.

Table 2. 10: Size and inference time results of per-channel affine quantization.

Quantization Schemes	Methods	Metrics			
		Size	Size Difference(%)	Inference Time	Inference Difference (%)
Per-channel Affine	<i>ResNet101</i>	43.66	74.41	16.7	76.42
	<i>RegNet-X-32GF</i>	106.91	74.66	63.5	57.66
	<i>Inceptionv3</i>	22.28	77.20	8.2	71.22
	<i>DenseNet-169</i>	14.14	72.26	22.5	54.08

Table 2. 11: Size and inference time results of per-channel symmetric quantization.

Quantization Schemes	Methods	Metrics			
		Size	Size Difference(%)	Inference Time	Inference Difference (%)
Per-channel Symmetric	<i>ResNet101</i>	43.66	74.41	16.3	77.10
	<i>RegNet-X-32GF</i>	106.91	74.66	65.2	56.53
	<i>Inceptionv3</i>	22.28	77.20	8.1	71.57
	<i>DenseNet-169</i>	14.14	72.26	22.6	53.87

Per-tensor quantization can make models smaller than per-channel quantization regardless of symmetric or affine configuration. However, the difference between these two methods is quite small approximately 0.6%, which is negligible. Therefore, we have considered the accuracy rates while making the decision. When the inference time of the models was compared, it was seen that the fastest model was Inceptionv3. Besides, all the models became faster, with at least a 50% improvement after employing quantization. In addition, accuracy values should also be considered when evaluating speed improvements. For example, the per-channel symmetric scheme speeded up ResNet101 by 77.1% while causing a 38.8% drop in accuracy. When we

examine all processes, although RegNet-X-32GF has a relatively large size and complex structure, it did not provide a significant advantage compared to other architectures. For this reason, it is more beneficial to use Inceptionv3 instead of RegNet-X-32GF, which shows similar behaviors. Besides, the second largest sized ResNet101, showed comparable results to DenseNet-169, but DenseNet-169 achieved slightly better performance with its smaller model size and fast inference time.

Finally, the best configurations for models were selected according to quantization and pruning results, and optimization was applied to all models by combining quantization and pruning. Pruned and quantized model performance results are presented in Table 2.12.

Table 2. 12: Performance evaluation results of quantized and pruned models.

Configurations	Methods	Metrics					
		Accuracy	Precision	Recall	F1 Score	Memory (MB)	Inference Time (ms)
Per-channel Affine + L1 Unstructured (0.6)	DenseNet-169	0.944	0.949	0.950	0.949	6.9	16.4
Per-tensor Affine + L1 Unstructured (0.7)	ResNet101	0.926	0.921	0.933	0.919	15.8	21.1
Per-tensor Symmetric + L1 Unstructured (0.3)	Inceptionv3	0.901	0.905	0.914	0.909	20.2	6.9
Per-tensor Affine + L1 Unstructured (0.7)	RegNet-X-32GF	0.921	0.915	0.944	0.929	67.8	55.5

L1 unstructured scheme, which is the most convenient technique for all models, is used for pruning. Then, the most successful quantization configurations are employed such as per-tensor affine quantization for ResNet101, per-tensor symmetric for Inceptionv3, per-tensor affine for RegNet-X-32GF and per-channel affine for DenseNet-169. According to Table 2.11, although a slight decrease in accuracy rate was observed, all optimized models achieved successful results with over 90% test accuracy. However,

DenseNet was the most resilient architecture against optimization, with only a 3.27% accuracy drop. When the memory gain of the pruned and quantized models was examined, there was a decrease in the memory footprint of approximately 75% in all models. This level of memory gain is significant for applications on low-power devices with limited memory. The main reason for this decrease is the reduction of numbers from 32-bit representation to 8-bit representation. Using an 8-bit-integer instead of a 32-bit float improves the performance of the model, saves power, makes operations faster and reduces produced heat on the devices.

When all the results are summarized, DenseNet-169 was the most suitable architecture to be optimized with L1 unstructured pruning and per-channel affine quantization schemes for the forest fire detection problem.



3. OBJECT DETECTION

3.1 Data Acquisition

Object detection methods are frequently used in applications where the object's location is essential, such as using UAVs for forest fire detection. This section presents to see the performance of the optimization methods given in the second chapter on object detection algorithms. Firstly, dataset selection is significant as the first step in object detection applications. Accordingly, private datasets are generally used in studies approaching this problem with the object detection method [60–62]. Since it is hard to distinguish flames and smoke in forest fire images, these datasets are not completely suitable for object detection algorithms. Hence, a new dataset was created by collecting aerial forest fire images from the Antalya Directorate General of Forestry. The dataset created consists of 994 forest fire images, and areas containing flame and smoke contained area were bounding boxed and labeled. Sample images from this dataset are shared in Figure 3.1.



Figure 3.1: Image samples from object detection dataset.

3.2 Object Detection Methods

Object detection is one of the common computer vision techniques which deals with identifying and locating objects of certain classes in the image. Object detection

generally is categorized into two categories single-stage detectors and multi-stage detectors. It can be summarized as Figure 3.2.

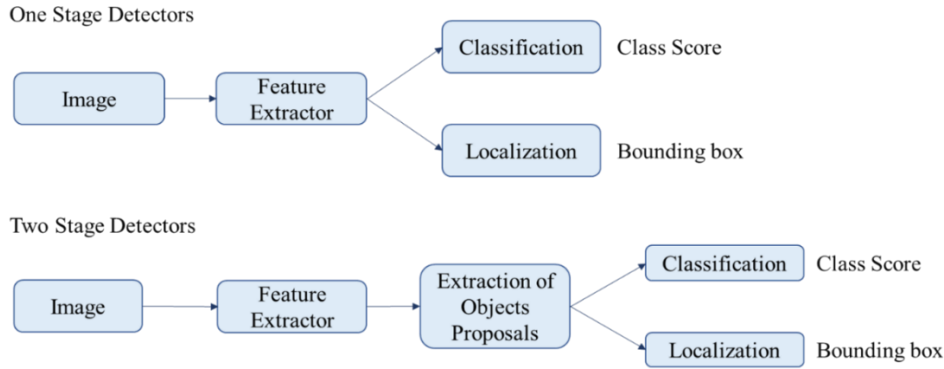


Figure 3.2: Object detection methods summary.

3.2.1 Faster R-CNN

Object detection methods classify in two different ways as single-stage detectors and multi-stage detectors. R-CNN-based algorithms are the most popular multi-stage detectors used in object detection applications. They provided more accurate detection compared to the other methods by using the regional proposal method. The first version R-CNN is a two-stage detector the first stage is responsible for determining regional proposals and the second stage is for classification task implementations [63]. The regional proposal method divides the image into approximately 2000 regions and decides which objects these regions belong to. Then, the selective search algorithm is used to extract regions by taking considered shape, size, texture, etc., of the objects in the image. After the region proposals are extracted, a feature vector is created from these regions. In the last step, SVM determines whether there is an object or a background in the region proposals. Although the Region proposal methods are successful, their cost and speed have been a drawback in real-time applications. The researchers made improvements to accelerate these architectures, considering the computational efficiency. To overcome mentioned drawbacks, R-CNN has been updated to Fast R-CNN by adding a layer that extracts equal-length feature vectors called the ROI pooling layer [64]. Since the improvements in Fast R-CNN did not reach the desired speed, Faster R-CNN was created by making new changes in the architecture as in Figure 3.3. Therefore, Faster R-CNN improved the region proposal method by using another CNN instead of the selective search algorithm for detecting the proposed regions to provide a faster and more accurate implementation [65].

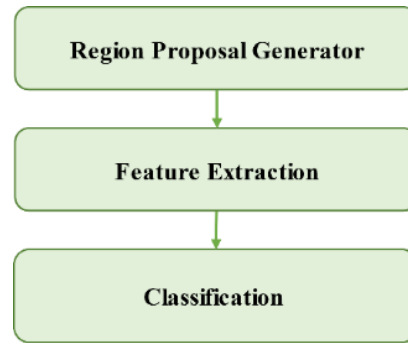


Figure 3.3: Faster R-CNN pipeline.

Input image fed into CNN and output of this network becomes feature map, and this map is used with the sliding window technique to generate anchor boxes. The bounding box classification and bounding box regressor methods are used to determine which of the anchor boxes is the object and which is the background. The bounding box classifier layer calculates the probabilities of whether the anchor boxes contain objects by calculating the intersection over union (IOU). The bounding box regressor layer learns the offsets for coordinates concerning the ground truth box for the anchor box that has been classified as foreground. Faster R-CNN provides a faster and more effective estimation with these improvements. The structure of this architecture can be briefly summarized in Figure 3.4.

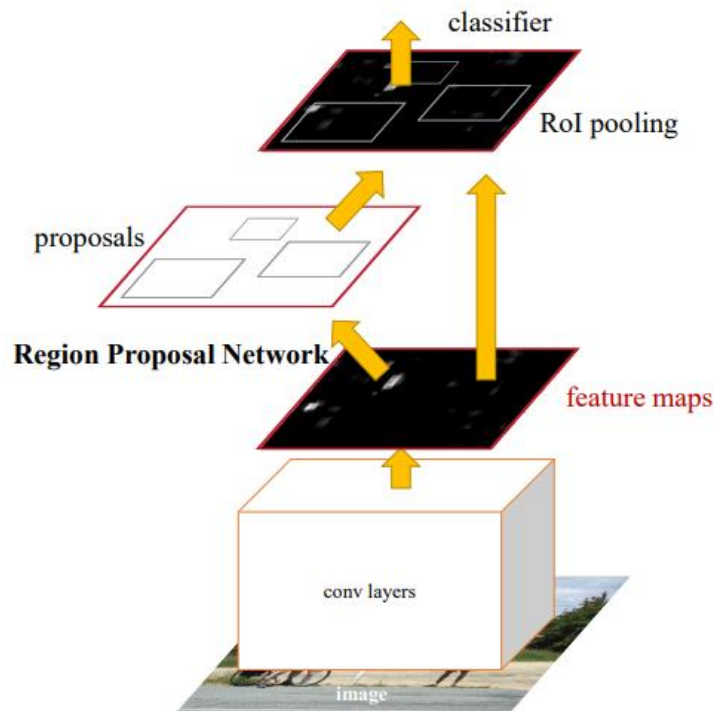


Figure 3.4: Faster R-CNN detection pipeline taken from [65].

3.2.2 YOLOv5

YOLO is a single-stage detector that uses neural networks to provide fast object detection. The first YOLO model was introduced by Redmond et. [66] in their 2015 paper titled “You Only Look Once” and has since become a popular approach for object detection tasks. Usually, a neural network model goes through both forward and backward propagation in order to predict, but YOLO undergoes only forward propagation to make the model run faster and consume fewer resources. This algorithm uses three main techniques residual blocks, bounding box regression, and IOU. The framework takes an input image and then divides this image into grid as shown in Figure 3.5. If an object is present in the cell, the cell is responsible for predicting a bounding box that tightly encloses the object.

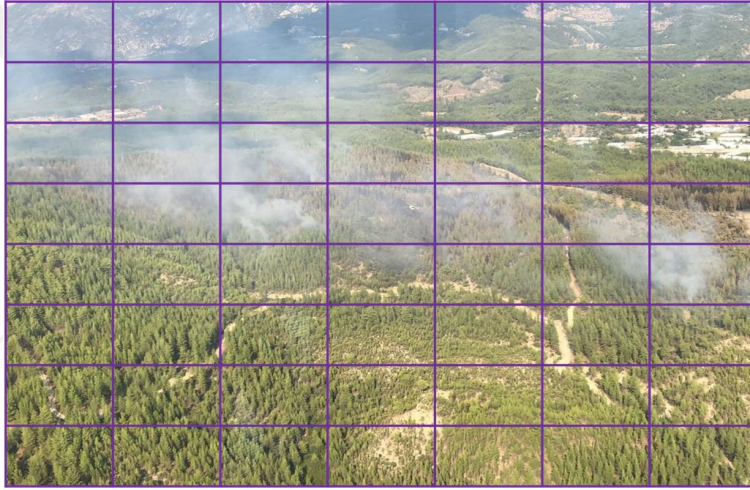


Figure 3.5: YOLOv5 grid method example.

Each grid is responsible for detecting objects by predicting bounding boxes with their confidence score. The confidence score is calculated to determine how reliable the prediction of object and bounding box coordinates is. These scores range from 0 to 1, from 0 to the lowest and from 1 to the highest. The output of each grid cell represented as equation 3.1:

$$y = (p_c, b_x, b_y, b_h, b_w, c) \quad (2.10)$$

where b_x and b_y are center and b_w width, b_h height of bounding box, p_c probability whether an object is present in the grid or not and c for represent classes as shown in Figure 3.6.

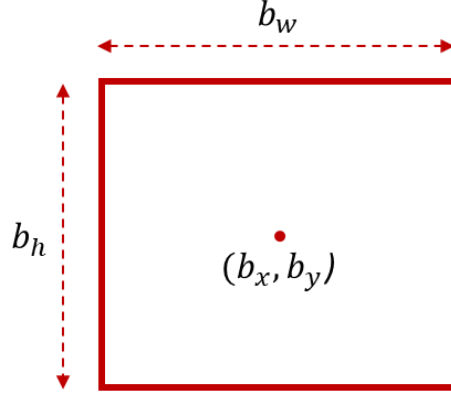


Figure 3.6: Bounding box representation.

In cases where the algorithm makes more than one prediction for the same class, the IOU method is applied with non-max suppression to determine the most accurate result. IOU is an evaluation metric that measures the accuracy of prediction. Figure 3.7 represents the implementation of the IOU method.

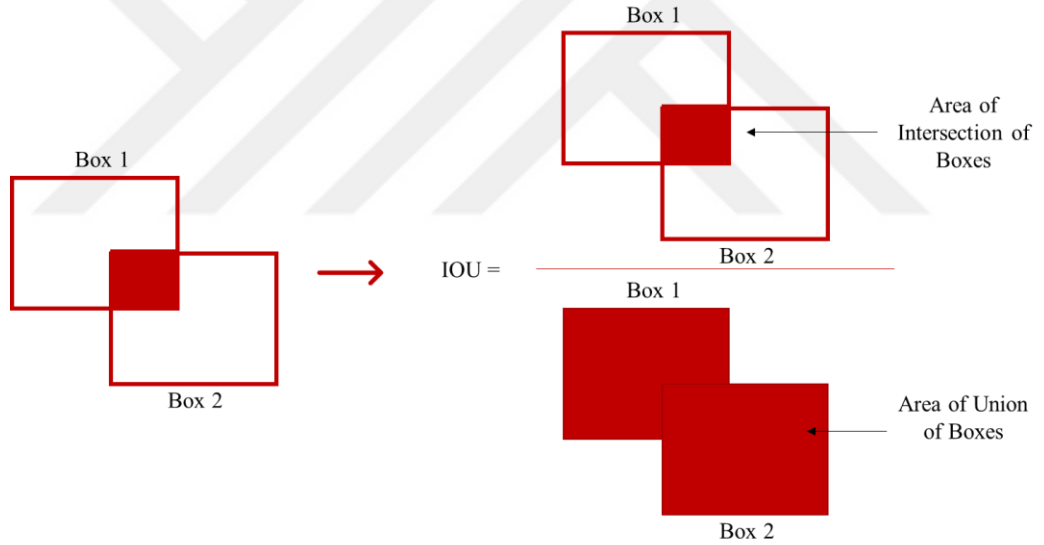


Figure 3.7: IOU calculation.

Non-maximum suppression is applied by taking the maximum value above the specified threshold of the IOU values calculated in the image. Thus, among the boxes determined for a single class, the most accurate one is selected. The YOLO architecture can be summarized with three main parts: backbone, neck, and head. The backbone determines features of the image at different granularities, the neck part combines image features and mixes them to forward to prediction, and the head applies prediction steps. In this study, YOLOv5 is used, which is the fifth developed version of YOLO.

3.2.3 Evaluation Metrics

The mean average precision is an evaluation technique for object detection applications. Evaluation of the object detection tasks consists of two essential concepts. The classification part controls the existing object in the image, and the localization part finds the location of the object in the image. Many datasets have non-uniform class distribution, and usage accuracy-based evaluation metrics can be deceptive for understanding the certainty of the model [67]. Therefore, the determination of the confidence score is essential to assess the risk of misclassifications. To understand the mAP, firstly, there is a need to explain average precision (AP), precision, recall, and IOU metrics. Precision calculates the ratio of true positives to total prediction true positives, and recall measures the proportion of true positives to ground truths. When a model has low precision but high recall, it predicts positive samples accurately but has many false positives. On the contrary, if a model has high precision but low recall, this indicates the model made correct prediction over only some of the positive samples. To get an insight from these metrics precision-recall curve is used to determine the best threshold value. When the curve is complex, the one way to simplify it is to measure the F1 score. F1 score shows the relationship between precision and recall. A lower F1 score indicates an imbalance between them. The AP calculation is a required step in choosing a reliable threshold. AP is measured as a calculation of the area under the precision-recall curve and represents this curve with a single value. Since the choice of IOU threshold is arbitrary, it is necessary to use the mAP method to standardize it. In the end, the mAP score is calculated by mean AP with overall thresholds with all the mentioned evaluation metrics as in Figure 3.8.

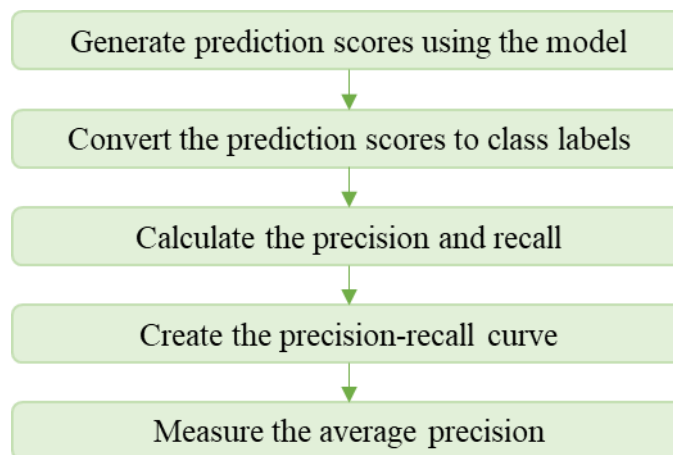


Figure 3.8: Object detection mAP score calculation steps.

3.3 Results and Discussion

Images collected from the Antalya Directorate General of Forestry were used to implement the object detection algorithms. Firstly, these collected images were prepared for algorithms. The fire regions were determined by drawing to the bounding box and labeled with image labeling tools. The labeling process is made in accordance with Faster R-CNN and YOLOv5. Faster R-CNN is compatible with PascalVOC-format, while YOLOv5 has its labeling format. In the labeling step, since smoke and flames are nested and cannot be easily distinguished in the images, only fire was determined as a class. Then, the dataset consisting of 944 images was prepared with only one class. To diversify the dataset, the augmentation method was applied by creating horizontal flip and -15, +15 rotated versions of the images as shown in Figure 3.9. After using data augmentation, the dataset had a total of 1560 images. Finally, the dataset has divided into %60 train, %20 validation, and %20 test set.

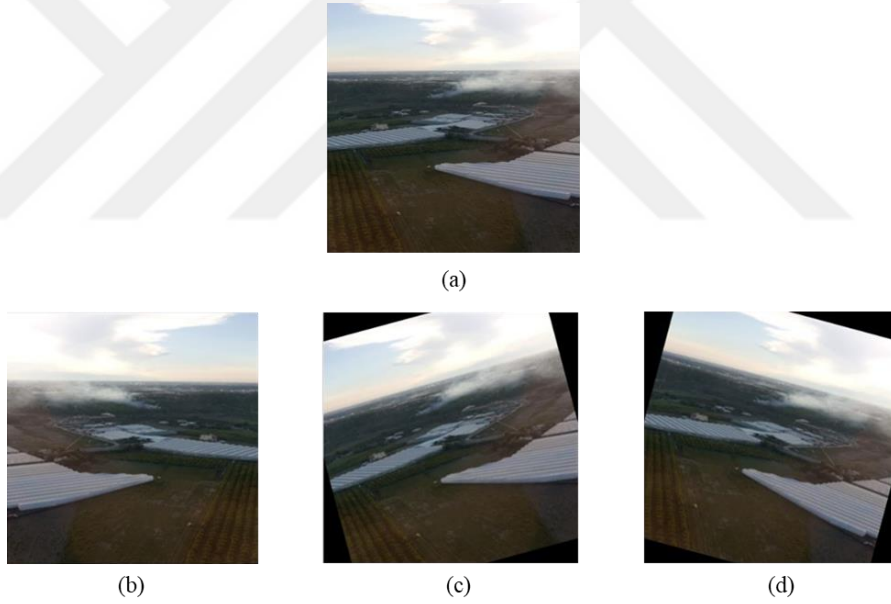


Figure 3.9: (a) Original image, (b) Horizontal flip of (a), (c) -15 rotated version of (a), (d) +15 rotated version of (a).

In order to observe the effect of optimization on object detection algorithms, Faster R-CNN was selected. The reason is that Faster R-CNN has capable detection structures that reach high accuracies with some drawbacks. One of the problems is that two-staged algorithms fall behind in terms of speed compared to one-staged algorithms. Thus, we aimed to overcome these drawbacks by using optimization techniques. In the end, we compared our results with state-of-the-art optimized YOLOv5 to understand our improvements because optimized YOLOv5 is commonly used in the literature

[68–70]. Firstly, DenseNet-169 was replaced with the backbone of the Faster R-CNN. Then both models were trained without adding optimization steps. During training, the total number of the predicted labels, precision, recall and mAP, and reserved memory were examined at the end of each epoch. Training and validation graphs are presented in Figure 3.10.

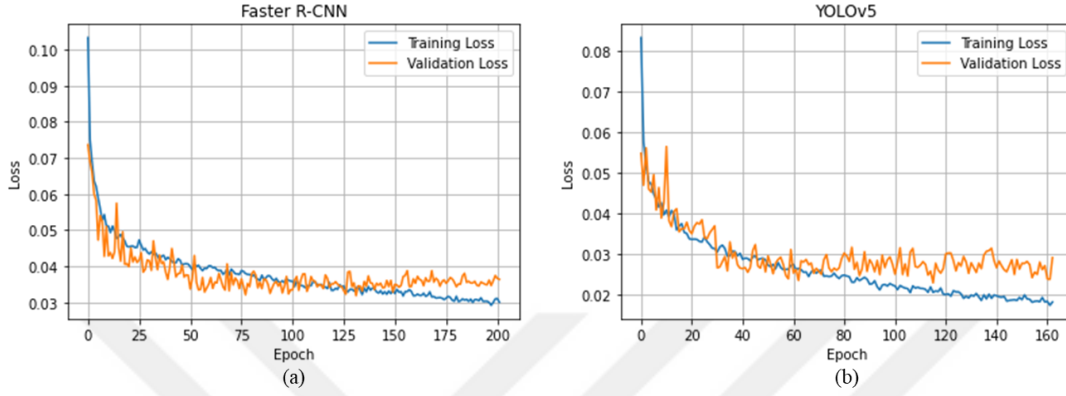


Figure 3.10: (a) Faster R-CNN training and validation loss graph, (b) YOLO5 training and validation graph.

When the graphs were examined, it was observed that the loss functions gradually decreased, and the training of the models was successful. Then, the trained models were evaluated using the test set to understand the generalization performance. mAP, precision, recall calculated and PR curve, and F1 Curve graphs were presented as the evaluate the success of the detector. Results are given in Figure 3.11, Figure 3.12 and Table 3.1 and Table 3.2.

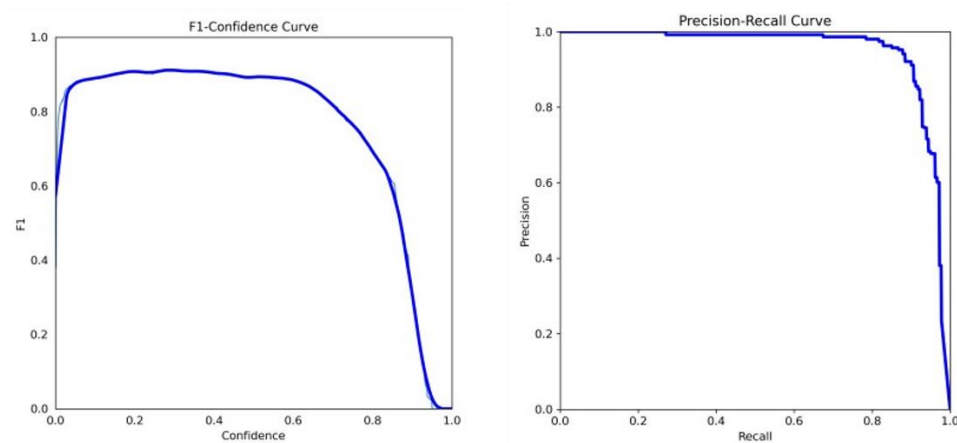


Figure 3.11: Faster R-CNN precision, recall, F1 and PR graphs.

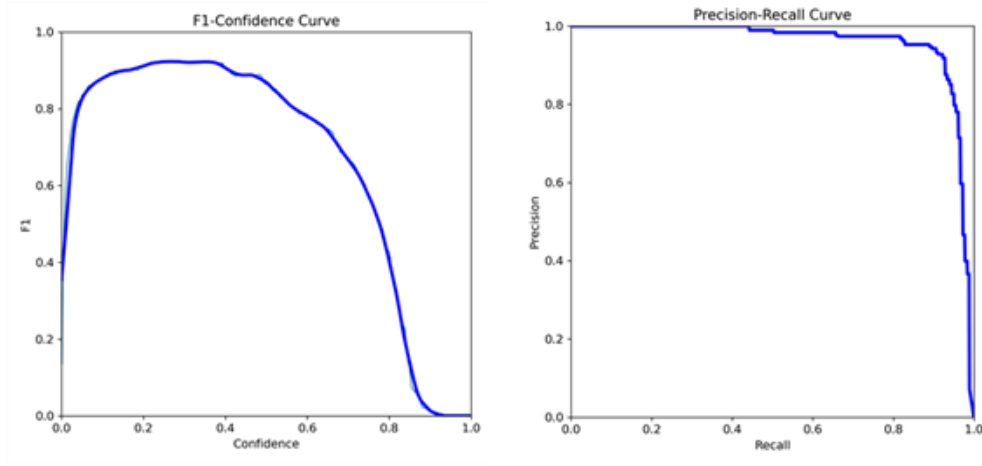


Figure 3.12: YOLOv5 precision, recall, F1 and PR graphs.

Table 3.1: Vanilla models evaluation results

Method	Precision	Recall	mAP
<i>Faster R-CNN</i>	0.897	0.874	0.891
<i>YOLOv5</i>	0.908	0.872	0.916

Table 3.2: Size and inference time of models

Method	Size (MB)	Inference Time (ms)
<i>Faster R-CNN</i>	159.74	65.7
<i>YOLOv5</i>	147.3	26.8

It has been observed that both models make successful predictions. The most challenging parts in forest fire detection are ambient sunlight reflections and dense clouds. Although fire smoke looks similar to the structure of clouds, there are differences. Since forest fires usually occur in summer, the color of the observed clouds is white. Depending on the material, fire smoke usually has a yellowish-white or gray tone. At the same time, the rays that the sun reflects on the camera create fire-like colors and shapes in the images. In order to make accurate predictions, the dataset must be fed into images containing these differences. Since the dataset we created consists of real forest fire images, it has been diversified to overcome these challenging points as much as possible. Thus, YOLOv5 achieved 0.916 mAP, and Faster R-CNN achieved 0.891 mAP scores. Even though the results were close, YOLOv5 performed slightly better. However, YOLOv5 has outperformed Faster R-CNN in terms of speed, as seen in Table 3.2. After the models were trained, optimization was implemented to make the models more efficient. The optimization configurations were selected

according to results in the previous chapter as per-channel affine quantization and L1 unstructured pruning for Faster R-CNN. Optimization with these configurations was implemented to Faster R-CNN's backbone. Then, state-of-the-art optimized YOLOv5 with default optimization settings was used to compare our optimized Faster R-CNN model. Optimized models' performance result shown in Table 3.3 and memory, inference time improvements in Table 3.4.

Table 3.3: Performance evaluation of optimized models.

Method	Precision	Recall	mAP
<i>Faster R-CNN</i>	0.882	0.868	0.871
<i>YOLOv5</i>	0.835	0.819	0.822

Table 3.4: Memory and inference time of optimized models.

Method	Size (MB)	Size Difference (%)	Inference Time (ms)	Inference Time Difference (%)
<i>Faster R-CNN</i>	112.43	29.61	34.3	47.99
<i>YOLOv5</i>	120.30	18.40	20.6	23.13

In the next step, we examined the advantages and disadvantages of optimizing these models. When we examined the optimization results, it was seen that pruning and quantization led to an accuracy drop in the models. This was the expected result because object detection architectures are more fragile due to their localization mechanisms. The accuracy drop rates of these two architectures are 5% for YOLOv5 and 3% for Faster R-CNN. Since the accuracy drop in YOLOv5 is more than Faster R-CNN, Faster R-CNN provided more reliable predictions. Moreover, both models became faster after the optimization. Although the increase in speed was more remarkable in Faster R-CNN, YOLOv5 still showed the best performance. In conclusion, according to the performances of the non-optimized models, the YOLOv5 was the fastest model with the highest mAP score. However, these results changed after the models were optimized. Faster R-CNN achieved a better mAP score with a smaller memory size compared to YOLOv5. Although YOLOv5 was faster than Faster R-CNN, Faster R-CNN has been more advantageous for forest fire detection, considering the mAP score and model size. Thus, choosing the best quantization and pruning methods for optimization provided better improvement in the model's performance.

4. CONCLUSIONS AND RECOMMENDATIONS

In this thesis, deep learning-based prediction models used in forest fire detection have been optimized to make them more compatible with low-power devices. Firstly, a dataset consisting 10,583 positive (fire) and 11,849 negative (no-fire) image samples are collected from the literature. Then, widely used deep learning architectures with proven predictive capabilities, i.e., ResNet101, RegNet-X-32GF, DenseNet-169 and Inceptionv3 are trained to detect and classify forest fire in these images. Despite their architectural differences, they all performed satisfactorily well by reaching over 95% accuracy. In the next step, these models are optimized by utilizing quantization and pruning techniques. For quantization; symmetric per-tensor, asymmetric per-tensor, symmetric per-channel, and asymmetric per-channel is applied. For pruning, L1 unstructured, random unstructured, L1 structured, random structured are applied with 0.1 to 0.9 compression ratios. Although a decrease in size and increase in inference speed was observed in all models, DenseNet-169 showed the greatest resiliency against accuracy drop compared to other models by reaching 94.4% accuracy, 86.45% memory saving, and 54.8% acceleration at inference time. Moreover, the best-performing quantization and pruning parameters are selected and applied for each model to push the limits of the architectures. DenseNet-169 still preserved the highest relative accuracy compared to other models while also having the lowest size. However, Inceptionv3 was shown to have the highest inference speed. Lastly, the image classification problem is extended to an object detection problem by utilizing Faster R-CNN model. DenseNet-169, as shown to be the most architecture compared to others, has been used as the backbone for the model. The resulting Faster R-CNN architecture has been trained to detect and localize the fire images. Moreover, a state-of-the-art object detection algorithm YOLOv5 is also trained on the same dataset to create a comparison baseline. Both models performed similarly and satisfactorily on the task by reaching 0.891 mAP for Faster R-CNN and 0.916 mAP for YOLOv5. Then, the Faster R-CNN model with DenseNet-169 backbone has been optimized by the most optimal quantization and pruning parameters as determined in the second section.

Yolov5, on the other hand, is also quantized and pruned by using the most optimal parameters proposed in the literature. When the optimized models are compared, Faster R-CNN showed higher accuracy and lower size compared to the Yolov5. However, Yolov5, as its most significant advantage over other object detection algorithms, had significantly faster inference time. When the results from both image classification and object detection sections are concerned, when the right optimization parameters are selected, DensetNet-169 shown to have not only the highest accuracy but also the lowest size among the compared methods. However, its inference time was higher compared to other inference time-oriented algorithms in the literature. However, the inference time difference depends on several factors such as camera speed and processing time of the images before feeding into the prediction models. Therefore, optimizing other time-consuming elements can still have a greater effect on the efficiency of low-power devices especially when the inference time difference between compared modes is concerned. Moreover, DenseNet-169 also has great potential for its use as a backbone for object detection algorithms and shown to reach even greater optimization metrics compared to the state-of-the-art. Besides the architectural comparisons, quantization and pruning methods with their respective variations have shown to be promising optimization arsenal for deep neural networks for the forest fire detection problem. However, there happens to be no “one-fit-for-all” approach and different variations showed varying performances on the employed architectures. Therefore, one needs to try different combinations to figure out specific optimal parameters for their architectures. Also, the combinational use of pruning and quantization is rather effective and increased the efficiency of the architecture.

We believe the optimization of powerful and highly complex deep learning architectures is the natural progression towards their use in low-power devices for detecting forest fires. And quantization and pruning techniques are shown to be useful tools in that regard that can pave the way for the mass adaptation of autonomous forest fire detection and reporting.

REFERENCES

- [1] **Ghorbanzadeh, O., Blaschke, T., Gholamnia, K., & Aryal, J.** (2019). Forest fire susceptibility and risk mapping using social/infrastructural vulnerability and environmental variables. *Fire*, 2(3), 50.
- [2] **Rodrigues, M., Gelabert, P. J., Ameztegui, A., Coll, L., & Vega-García, C.** (2021). Has COVID-19 halted winter-spring wildfires in the Mediterranean? Insights for wildfire science under a pandemic context. *Science of the total environment*, 765, 142793.
- [3] **Yu, L., Wang, N., & Meng, X.** (2005, September). Real-time forest fire detection with wireless sensor networks. In *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.* (Vol. 2, pp. 1214-1217). IEEE.
- [4] **Zhang, J., Li, W., Han, N., & Kan, J.** (2008). Forest fire detection system based on a ZigBee wireless sensor network. *Frontiers of Forestry in China*, 3(3), 369-374.
- [5] **Tsao, S. L., & Huang, C. H.** (2011). A survey of energy efficient MAC protocols for IEEE 802.11 WLAN. *Computer Communications*, 34(1), 54-67.
- [6] **Chen, T. H., Kao, C. L., & Chang, S. M.** (2003, October). An intelligent real-time fire-detection method based on video processing. In *IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, 2003. Proceedings.* (pp. 104-111). IEEE.
- [7] **Horng, W. B., Peng, J. W., & Chen, C. Y.** (2005, March). A new image-based real-time flame detection method using color analysis. In *Proceedings. 2005 IEEE Networking, Sensing and Control, 2005.* (pp. 100-105). IEEE.
- [8] **Marbach, G., Loepfe, M., & Brupbacher, T.** (2006). An image processing technique for fire detection in video images. *Fire safety journal*, 41(4), 285-289.
- [9] **Liu, C. B., & Ahuja, N.** (2004, August). Vision based fire detection. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* (Vol. 4, pp. 134-137). IEEE.
- [10] **Dedeoglu, N., Toreyin, B. U., Gudukbay, U., & Cetin, A. E.** (2005, March). Real-time fire and flame detection in video. In *Proceedings.*

(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. (Vol. 2, pp. ii-669). IEEE.

- [11] **Töreyn, B. U., Dedeoğlu, Y., Güdükbay, U., & Cetin, A. E.** (2006). Computer vision based method for real-time fire and flame detection. *Pattern recognition letters*, 27(1), 49-58.
- [12] **Zhang, Z., Zhao, J., Zhang, D., Qu, C., Ke, Y., & Cai, B.** (2008, December). Contour based forest fire detection using FFT and wavelet. In 2008 International conference on computer science and software engineering (Vol. 1, pp. 760-763). IEEE.
- [13] **Qi, X., & Ebert, J.** (2009). A computer vision based method for fire detection in color videos. *International journal of imaging*, 2(S09), 22-34.
- [14] **Chi, R., Lu, Z. M., & Ji, Q. G.** (2017). Real-time multi-feature based fire flame detection in video. *IET Image Processing*, 11(1), 31-37.
- [15] **Kong, S. G., Jin, D., Li, S., & Kim, H.** (2016). Fast fire flame detection in surveillance video using logistic regression and temporal smoothing. *Fire Safety Journal*, 79, 37-43.
- [16] **Gomez-Rodriguez, F., Arrue, B. C., & Ollero, A.** (2003, September). Smoke monitoring and measurement using image processing: application to forest fires. In *Automatic Target Recognition XIII* (Vol. 5094, pp. 404-411). SPIE.
- [17] **Töreyn, B. U., Dedeoğlu, Y., & Cetin, A. E.** (2005, September). Wavelet based real-time smoke detection in video. In 2005 13th European signal processing conference (pp. 1-4). IEEE.
- [18] **Chen, T. H., Yin, Y. H., Huang, S. F., & Ye, Y. T.** (2006, December). The smoke detection for early fire-alarming system base on video processing. In 2006 international conference on intelligent information hiding and multimedia (pp. 427-430). IEEE.
- [19] **Yang, J., Chen, F., & Zhang, W.** (2008, October). Visual-based smoke detection using support vector machine. In 2008 Fourth International Conference on Natural Computation (Vol. 4, pp. 301-305). IEEE.
- [20] **Ma, L., Wu, K., & Zhu, L.** (2010, October). Fire smoke detection in video images using Kalman filter and Gaussian mixture color model. In 2010 International Conference on Artificial Intelligence and Computational Intelligence (Vol. 1, pp. 484-487). IEEE.
- [21] **Tung, T. X., & Kim, J. M.** (2011). An effective four-stage smoke-detection algorithm using video images for early fire-alarm systems. *Fire Safety Journal*, 46(5), 276-282.
- [22] **Kim, S., Lee, W., Park, Y. S., Lee, H. W., & Lee, Y. T.** (2016, December). Forest fire monitoring system based on aerial image. In 2016 3rd

international conference on information and communication technologies for disaster management (ICT-DM) (pp. 1-6). IEEE.

- [23] **Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L.** (2009). ImageNet: A Large-Scale Hierarchical Image Database, in ‘CVPR09’. IEEE Computer Society, Miami, Florida, USA, 248-255.
- [24] **Gonzalez, A., Zuniga, M. D., Nikulin, C., Carvajal, G., Cardenas, D. G., Pedraza, M. A. & Akhloufi, M. A.** (2017). Accurate fire detection through fully convolutional network.
- [25] **Toulouse, T., Rossi, L., Celik, T., & Akhloufi, M.** (2016). Automatic fire pixel detection using image processing: a comparative analysis of rule-based and machine learning-based methods. *Signal, Image and Video Processing*, 10(4), 647-654.
- [26] **Hüttner, V., Steffens, C. R., & da Costa Botelho, S. S.** (2017, November). First response fire combat: Deep leaning based visible fire detection. In 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR) (pp. 1-6). IEEE.
- [27] **Chenebert, A., Breckon, T. P., & Gaszczak, A.** (2011, September). A non-temporal texture driven approach to real-time fire detection. In 2011 18th IEEE international conference on image processing (pp. 1741-1744). IEEE.
- [28] **Steffens, C. R., Botelho, S. S. D. C., & Rodrigues, R. N.** (2016, October). A texture driven approach for visible spectrum fire detection on mobile robots. In 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR) (pp. 257-262). IEEE.
- [29] **Shen, D., Chen, X., Nguyen, M., & Yan, W. Q.** (2018, April). Flame detection using deep learning. In 2018 4th International conference on control, automation and robotics (ICCAR) (pp. 416-420). IEEE.
- [30] **Zhang, Q. X., Lin, G. H., Zhang, Y. M., Xu, G., & Wang, J. J.** (2018). Wildland forest fire smoke detection based on faster R-CNN using synthetic smoke images. *Procedia engineering*, 211, 441-446.
- [31] **Aslan, S., Güdükbay, U., Töreyn, B. U., & Cetin, A. E.** (2019). Deep convolutional generative adversarial networks based flame detection in video. *arXiv preprint arXiv:1902.01824*.
- [32] **Goel, A., Tung, C., Lu, Y. H., & Thiruvathukal, G. K.** (2020, June). A survey of methods for low-power deep learning and computer vision. In 2020 IEEE 6th World Forum on Internet of Things (WF-IoT) (pp. 1-6). IEEE.
- [33] **Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B. & Hua, X. S.** (2019). Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7308-7316).

- [34] **Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Gutttag, J.** (2020). What is the state of neural network pruning?. *Proceedings of machine learning and systems*, 2, 129-146.
- [35] **Alyamkin, S., Ardi, M., Berg, A. C., Brighton, A., Chen, B., Chen, Y., ... & Zhuo, S.** (2019). Low-power computer vision: Status, challenges, and opportunities. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2), 411-421.
- [36] **Foggia, P., Saggese, A., & Vento, M.** (2015). Real-time fire detection for video-surveillance applications using a combination of experts based on color, shape, and motion. *IEEE TRANSACTIONS on circuits and systems for video technology*, 25(9), 1545-1556.
- [37] **Toulouse, T., Rossi, L., Celik, T., & Akhloufi, M.** (2016). Automatic fire pixel detection using image processing: a comparative analysis of rule-based and machine learning-based methods. *Signal, Image and Video Processing*, 10(4), 647-654.
- [38] **Weber, E., Papadopoulos, D. P., Lapedriza, A., Ofli, F., Imran, M., & Torralba, A.** (2022). Incidents1M: a large-scale dataset of images with natural disasters, damage, and incidents. *arXiv preprint arXiv:2201.04236*.
- [39] **Sesver, D., Gençoğlu, A. E., Yıldız, Ç. E., Günindi, Z., Habibi, F., Yazıcı, Z. A., & Ekenel, H. K.** (2022). VIDI: A Video Dataset of Incidents. *arXiv preprint arXiv:2205.13277*.
- [40] **Shinohara, R. T., Sweeney, E. M., Goldsmith, J., Shiee, N., Mateen, F. J., Calabresi, P. A., ... & Alzheimer's Disease Neuroimaging Initiative.** (2014). Statistical normalization techniques for magnetic resonance imaging. *NeuroImage: Clinical*, 6, 9-19.
- [41] **O'Shea, K., & Nash, R.** (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
- [42] **Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S.** (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187, 27-48.
- [43] **Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T.** (2018). Recent advances in convolutional neural networks. *Pattern recognition*, 77, 354-377.
- [44] **Feng, J., He, X., Teng, Q., Ren, C., Chen, H., & Li, Y.** (2019). Reconstruction of porous media from extremely limited information using conditional generative adversarial networks. *Physical Review E*, 100(3), 033308.

- [45] **He, K., Zhang, X., Ren, S., & Sun, J.** (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- [46] **Wu, Z., Shen, C., & Van Den Hengel, A.** (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90, 119-133.
- [47] **Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q.** (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
- [48] **Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., & Keutzer, K.** (2014). Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*.
- [49] **Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A.** (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
- [50] **Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z.** (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
- [51] **Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., & Dollár, P.** (2020). Designing network design spaces. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10428-10436).
- [52] **Xu, J., Pan, Y., Pan, X., Hoi, S., Yi, Z., & Xu, Z.** (2022). RegNet: self-regulated network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*.
- [53] **Weiss, K., Khoshgoftaar, T. M., & Wang, D.** (2016). A survey of transfer learning. *Journal of Big data*, 3(1), 1-40.
- [54] **Krishnamoorthi, R.** (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.
- [55] **Wu, H., Judd, P., Zhang, X., Isaev, M., & Micikevicius, P.** (2020). Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*.
- [56] **Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K.** (2021). A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*.

- [57] **Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Kalenichenko, D.** (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).
- [58] **Frankle, J., & Carbin, M.** (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635.
- [59] **Paganini, M., & Forde, J.** (2020). On iterative neural network pruning, reinitialization, and the similarity of masks. arXiv prep
- [60] **Wu, S., & Zhang, L.** (2018, December). Using popular object detection methods for real time forest fire detection. In 2018 11th International symposium on computational intelligence and design (ISCID) (Vol. 1, pp. 280-284). IEEE.
- [61] **Hu, Y., Zhan, J., Zhou, G., Chen, A., Cai, W., Guo, K., ... & Li, L.** (2022). Fast forest fire smoke detection using MVMNet. Knowledge-Based Systems, 241, 108219.
- [62] **Li, S., Yan, Q., & Liu, P.** (2020). An efficient fire detection method based on multiscale feature extraction, implicit deep supervision and channel attention mechanism. IEEE Transactions on Image Processing, 29, 8467-8475.
- [63] **Girshick, R., Donahue, J., Darrell, T., & Malik, J.** (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).
- [64] **Girshick, R.** (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
- [65] **Ren, S., He, K., Girshick, R., & Sun, J.** (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.
- [66] **Redmon, J., Divvala, S., Girshick, R., & Farhadi, A.** (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
- [67] **Padilla, R., Netto, S. L., & Da Silva, E. A.** (2020, July). A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP) (pp. 237-242). IEEE.
- [68] **Cui, P., Zhang, J., Han, B., & Wu, Y.** (2022, February). Performance evaluation and model quantization of object detection algorithm for infrared image. In Seventh Asia Pacific Conference on Optics Manufacture and

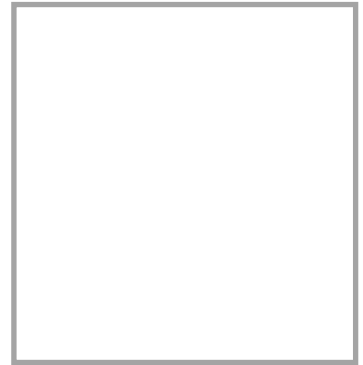
2021 International Forum of Young Scientists on Advanced Optical Manufacturing (APCOM and YSAOM 2021) (Vol. 12166, pp. 554-559). SPIE.

- [69] **Fang, J., Liu, Q., & Li, J.** (2021, April). A deployment scheme of YOLOv5 with inference optimizations based on the Triton inference server. In 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA) (pp. 441-445). IEEE.
- [70] **Zhang, J., Wang, P., Zhao, Z., & Su, F.** (2021, September). Pruned-YOLO: Learning efficient object detector using model pruning. In International Conference on Artificial Neural Networks (pp. 34-45). Springer, Cham.





CURRICULUM VITAE



Name Surname : Berrin SAVDA

EDUCATION:

- **B.Sc.** : 2019, Izmir Katip Celebi University, Faculty of Engineering and Architecture, Electrical and Electronics Engineering
- **M.Sc.** : 2023, Istanbul Technical University, Graduate School, Department of Electronics and Communication Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2021-Current, Software Engineer, Vodafone A.Ş
- 2019, Completed B.Sc. at Izmir Katip Celebi University Electrical and Electronics Engineering Department with high honor ranking 2nd in department.