

T.C.
BAHCESEHIR UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
THE DEPARTMENT OF INDUSTRIAL ENGINEERING

PRUNING THE ENSEMBLE OF DEEP NEURAL NETWORKS
WITH SECOND-ORDER CONE PROGRAMMING

PHD THESIS
BUSE ÇİSİL GÜLDOĞUŞ

ISTANBUL 2024

T.C.
BAHÇEŞEHİR UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
THE DEPARTMENT OF INDUSTRIAL ENGINEERING

PRUNING THE ENSEMBLE OF DEEP NEURAL NETWORKS
WITH SECOND-ORDER CONE PROGRAMMING

PHD THESIS
BUSE ÇİSİL GÜLDOĞUŞ

THESIS ADVISOR
PROF. DR. SÜREYYA AKYÜZ

ISTANBUL 2024



T.C.
BAHÇEŞEHİR UNIVERSITY
GRADUATE SCHOOL
...../...../.....
PhD THESIS APPROVAL FORM

Name Surname	Buse Çisil Göldoğan
Student Number	1615752
Program Name	Industrial Engineering
Title of Thesis	Pruning The Ensemble of Deep Neural Networks with Second-Order Cone Programming
Thesis Defense Date	18/01/2024

It has been approved by the Graduate School that this thesis has fulfilled the necessary conditions as a PhD thesis.

.....
**Director of
Graduate
School**

This Thesis has been read by us, it has been deemed sufficient and accepted as a PhD thesis in terms of quality and content.

PhD Thesis Defense Jury			
Thesis Defense Jury	Title - Name / Surname	Institution	Signature
Thesis Advisor	Prof. Dr. Süreyya Akyüz	Bahcesehir University	
Member of Thesis Monitoring Committee	Prof. Dr. Sabri Tankut Atan	Bahcesehir University	
Member of Thesis Monitoring Committee (outside institution)	Prof. Dr. Birsan Eygi Erdoğan	Marmara University	
Member	Asst. Prof. Dr. Ali Hamidoğlu	Bahcesehir University	
Member (outside institution)	Assoc. Prof. Dr. Burcu Tunga	Istanbul Technical University	



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Buse isil Gldoğuş

Signature:

ABSTRACT

PRUNING THE ENSEMBLE OF DEEP NEURAL NETWORKS WITH SECOND-ORDER CONE PROGRAMMING

Güldoğuş, Buse Çisil

Industrial Engineering PhD Program

Thesis Advisor: Prof. Dr. Süreyya Akyüz

January 2024, 86 pages

Ensemble methods hold high significance in machine learning as they combine multiple models to create an optimal predictive model. Ensemble pruning reduces the computational complexity of ensemble models by selecting a subset of classifiers. This subset selection involves a trade-off between accuracy and diversity. Thus, this thesis focuses on a novel mathematical model to prune the ensemble of Convolutional Neural Networks (CNNs), employing sparse second-order cone programming in which accuracy and diversity tradeoff is optimized. Experiments conducted on CIFAR-10, CIFAR-100, and MNIST datasets showcase promising results by substantially reducing the number of models while maintaining performance metrics. The proposed model is further extended to an embedded feature selection algorithm. It functions by training a neural network on individual features within a dataset, utilizing their respective outputs to construct a probability class distribution alongside predictions. This methodological approach integrates second-order conic programming to identify features that significantly contribute to precise classification using neural networks as embedded classifiers.

Key Words: Pruning, CNN, SOCP, Ensemble, Feature Selection

ÖZET

İKİNCİ DERECEDEN KONİK PROGRAMLAMA İLE DERİN SİNİR AĞLARI TOPLULUĞUNUN BUDANMASI

Güldoğuş, Buse Çisil

Endüstri Mühendisliği Doktora Programı

Tez Danışmanı: Prof. Dr. Süreyya Akyüz

Ocak 2024, 86 sayfa

Topluluk yöntemleri, birden fazla modeli birleştirerek optimal bir tahmin modeli oluşturması açısından makine öğreniminde büyük önem taşımaktadır. Topluluk budama, topluluk modellerinin hesaplama karmaşıklığını bir sınıflandırıcı alt kümesi seçerek azaltır. Bu alt küme seçimi, doğruluk ve çeşitlilik arasında bir denge gerektirir. Bu tez, farklı derinlik ve katmanlara sahip Evrisimli Sinir Ağları'nın (CNN'ler) topluluğunu seyrek ikinci derece koni programlamayı kullanarak budamak için yeni bir matematiksel model üzerinde odaklanmaktadır. Bu yaklaşım aynı zamanda doğruluk ve çeşitlilik dengelemesini optimize etmektedir. CIFAR-10, CIFAR-100 ve MNIST veri setlerinde yapılan deneyler, performans metrikleri korunurken model sayısının önemli ölçüde azaltıldığını göstermektedir. Önerilen model, gömülü bir özellik seçimi algoritması olarak kullanılmıştır. Bu, veri seti içindeki öznelikleri eğiten bir sinir ağı ile işlev görür ve bunların ilgili çıktıları kullanarak tahminlerle birlikte bir olasılık sınıf dağılımı oluşturur. Bu metodolojik yaklaşım, sinir ağlarını gömülü sınıflandırıcılar olarak kullanarak sınıflandırmaya önemli katkıda bulunan öznelik belirlemek için ikinci derece koni programlamayı entegre etmektedir.

Anahtar Kelimeler: Budama, CNN, SOCP, Topluluk, Öznelik Seçimi

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Prof. Dr. Süreyya Akyüz, whose guidance, encouragement, and unwavering support have been invaluable throughout this research journey. Your expertise, mentorship, and dedication have been instrumental in shaping this thesis and my academic growth.

I am profoundly grateful to the members of my PhD thesis monitoring committee; Prof. Dr. Sabri Tankut Atan and Prof. Dr. Birsen Eygi Erdoğan for their insightful feedback, valuable suggestions, and critical evaluation. Their expertise and constructive criticisms have significantly enriched the quality of this work.

Special thanks to my colleagues, Muhammad Ammar Ali and Abdullah Nazhat Abdullah, for their constant support, fruitful discussions, and collaborative efforts. Your contributions, ideas, and encouragement have been a source of inspiration, fostering a stimulating academic environment.

Lastly, I am indebted to my family for their unwavering love, understanding, and encouragement throughout this academic pursuit. Your support has been the cornerstone of my success.

This thesis is funded by The Scientific and Technological Research Council of Turkey (TUBITAK) under the project number 119E100.

Buse Çisil GÜLDOĞUŞ
İstanbul, 2024

To My Family



TABLE OF CONTENTS

ETHICAL CONDUCT	iii
ABSTRACT.....	iv
ÖZET v	
ACKNOWLEDGEMENTS	vi
LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF SYMBOLS/ABBREVIATIONS	viii
Introduction.....	1
Literature Review.....	10
2.1 Optimizing Neural Networks: Deep Learning Network Pruning, Crafting Deep Ensembles, and Fine-Tuning Ensemble Pruning for Efficient AI Models.....	11
2.2 Unlocking Efficiency: Sparse Optimization Models and the Power of Second-Order Cone Programming in Pruning AI Models.....	22
Methods	36
3.1 Creating the Optimization Model that Satisfies the Most Suitable Conditions from the Library	39
3.1.1 Shannon entropy	41
3.1.2 Second-order cone programming.....	43
3.1.2.1 <i>Duality and complementarity slackness.</i>	45
3.2 Redefining the proposed loss function as Second-Order Cone Programming.....	49
3.3 A New Feature Selection Technique: Feature Selection via Second-Order Cone Programming – FSOCP	58
3.3.1 Background materials.....	59
3.3.1.1 <i>Similarity based methods</i>	61
i. Fisher Score.....	62
ii. Trace Ratio Criterion (Supervised)	62
iii. ReliefF (Supervised)	63
3.3.1.2 <i>Information theoretical based methods</i>	63
i. MIM-Mutual Information Maximization (or Information Gain).....	65
ii. MIFS - Mutual Information Feature Selection.....	66
iii. MRMR-Minimum Redundancy Maximum Relevance.....	66
iv. CIFE - Conditional Infomax Feature Extraction.....	67
v. JMI- Joint Mutual Information	67
vi. CMIM-Conditional Mutual Information Maximization	68
vii. DISR-Double Input Symmetrical Relevance	68

viii.	FCBF-Fast Correlation Based Filter	69
i.	Interaction Capping-ICAP	69
3.3.1.3	<i>Sparse learning based methods</i>	69
3.3.1.4	<i>Statistical based methods</i>	70
i.	F-Score (Supervised).....	71
ii.	Gini Index (Supervised)	72
iii.	CFS (Supervised)	72
iv.	Chi-Square (Supervised)	72
3.3.1.5	<i>Wrapper methods</i>	73
i.	Sequential Backward Selection Algorithm Based on Support Vector Machine and Decision Tree	73
3.3.2	Feature selection via second-order cone programming.	74
Results		78
4.1	Results of Pruning the Ensemble of CNNs with SOCP	78
4.2	Results of FSOCP	80
Discussions		84
5.1	Discussions and Future Outlook for Pruning the Ensemble of CNNs with SOCP	84
5.2.	Discussions and Future Outlook on Feature Selection by using SOCP.	85
REFERENCES		87

LIST OF TABLES

TABLES

Table 1 Dataset and Their Definitions	75
Table 2 Performance Result of Full and Pruned Ensemble with SOCP (Güldoğuş, Abdullah, & Ali, 2023)	78
Table 3 Best, average and worst performance results of CIFAR-10, CIFAR-100 and MNIST	79
Table 4. Performances of the proposed FSOCP algorithm and other feature selection techniques in the literature	81
Table 5. Running times of feature selection algorithms	82



LIST OF FIGURES

FIGURES

Figure 1 The general structure of RNP (Lin, Rao, Lu, & Zhou, 2017).....	13
Figure 2 Performance results due to selection of different norms (Han, Mao, & Dally, 2016).	14
Figure 3 The steps for pruning convolutional filters in deep networks (Jordao, Kloss, Yamada, & Schwartz, 2018).	16
Figure 4 The state of neural networks before and after pruning (Liu, Xin, Mu, & Zhu, 2019).	20
Figure 5 A structured sequence of convex optimization.....	33
Figure 6 The flowchart of the FSOCP algorithm.....	76



LIST OF SYMBOLS/ABBREVIATIONS

ℓ_0	Zero-Norm
ℓ_1	One-Norm
ℓ_2	Euclidean Norm
w_i	Weight of the i_{th} classifier
f_{ens}	Consensus function of an ensemble
f_i^1	Probability of i_{th} sample
p_i	Probability of output distributions of i_{th} classifier
f_{gt}	Function of Ground truth
L_{ens}	Loss function of an ensemble
α	Regularization constant for accuracy-diversity tradeoff
C	Number of classes
$H(A)$	Shannon Entropy of a function A
x, b, d	Optimization variables
A, c, R	Matrices
A^T	Transpose of matrix A
ℓ_k	k -dimensional unit
n_i	Dimension of second-order cone constraint
m	Number of rows in matrix A
$\mathbf{x} = (x_i; \dots x_r)$	A vector
$\mathbf{y}, \mathbf{z}$	Dual feasible vectors
Q	A cone
<u>P</u>	Primal problem
<u>D</u>	Dual problem
$\hat{\mathbf{x}}$	Strict feasible point
$int(Q)$	Interior of cone Q
$span(A^T)$	Set of all linear combinations of the vectors in transpose of matrix A
P	Orthogonal projection operator
A^+	Pseudo inverse of matrix A
t	A quadratic term
Q_{rot}	A rotated cone
λ	A regularization constant
$f(x)$	A function
$\ x\ _0$	The zero-norm of x
$\ x\ _1$	The one-norm of x

u	Bound variable for relaxation of zero-norm
K	A convex cone
L	A lower triangular matrix
D	A dataset
M	Number of models of Convolutional Neural Networks
b	Convolutional blocks
v	Convolutional filters
d	Drop-out layer percentage
c	Pooling mechanism
s	Fully connected layer size
r	Learning rate
h	Sensitivity threshold
e	Experiment size
P	Accuracy of pruned ensemble
N	Number of samples
f	A feature
S	Affinity matrix
$\hat{S}$	Similarity matrix
σ	Standard deviation
μ	Mean
σ^2	Variance
S_w	Intra-class matrix
S_b	Inter-class matrix
D_w	Diagonal matrix
L_w	Laplacian matrix
W	Indicator matrix
F	Number of features in a dataset
$d(\cdot)$	Euclidean distance matrix
$I(X; Y)$	Information gain between variable X and Y
$J(\cdot)$	Feature selection criterion
$g(\cdot)$	Function of two variables
β, λ	Non-negative parameter
SU	Symmetric uncertainty
C_s	Class label s
$p(\cdot)$	Probability
r_{ff}	The average feature-feature intercorrelation
n_m	A random neural network model

t	Threshold for pruning
B_r	Best experiment
A_r	Average experiment
W_r	Worst experiment
k	Number of selected features
AI	Artificial Intelligence
ML	Machine Learning
GPU	Graphics Processing Units
CNNs	Convolutional Neural Networks
RNNs	Recurrent Neural Network
ANNs	Artificial Neural Networks
LSTMs	Long Short-Term Memory Networks
GRU	Gated Recurrent Units
NP	Non-deterministic Polynomial Time
SDP	Semi-Definite Programming
DC	Difference of Convex
SOCp	Second-Order Cone Programming
FSOCp	Feature Selection via Second-Order Cone Programming
RNP	Runtime Neural Pruning
MDP	Markov Decision Process
PLS	Partial Least Squares
VIP	Variable Importance in Projection
IoT	Internet of Things
SGD	Stochastic Gradient Descent
DNNs	Deep Neural Networks
MKL	Multiple Kernel Learning
SVM	Support Vector Machines
LP	Linear Programming
QP	Quadratic Programming
PMKL	Parallel Multiple Kernel Learning
ADMM	Alternating Direction Method of Multipliers
SMKL	Stacked Generalization Multiple Kernel Learning
LDA	Linear Discriminant Analysis
ANOVA	Analysis of Variance

PCA	Principal Component Analysis
SPCA	Sparse Principal Component Analysis
OMP	Orthogonal Matching Pursuit
QCQP	Quadratically Constrained Quadratic Programming
CP	Convex Programming
ReLU	Rectified Linear Unit
MIM	Mutual Information Maximization
MIFS	Mutual Information Feature Selection
MRMR	Minimum Redundancy Maximum Relevance
CIFE	Conditional Infomax Feature Extraction
JMI	Joint Mutual Information
CMIM	Conditional Mutual Information Maximization
DISR	Double Input Symmetrical Relevance
FCBF	Fast Correlation Based Filter
ICAP	Interaction Capping
RFS	$l_{2,1}$ -norm based feature selection algorithm
CFS	Correlation-based Feature Selection
SBS	Sequential Backward Selection
DT	Decision Tree
SFS	Sequential Forward Selection

Chapter 1

Introduction

Artificial Intelligence (AI) has risen to the forefront of our contemporary society, firmly establishing itself as a transformative force, largely attributable to remarkable technological advancements. These advancements have catalyzed AI's rapid ascent, permeating every aspect of modern life. Within the expansive domain of AI, Machine Learning (ML) has emerged as a pivotal subfield, dedicated to the development of predictive models through a diverse array of algorithms. Machine learning serves as a vital component, emphasizing the development of predictive models through the utilization of diverse algorithms.

AI, as a comprehensive field, encompasses a wide spectrum of subdomains, each with a unique focus and set of capabilities. Among these, Machine Learning stands out as one of the most influential and practical branches. It operates on the fundamental premise that machines can be designed to learn from data, adapt to evolving circumstances, and make informed predictions or decisions. ML algorithms, ranging from simple linear regression to complex ensemble models, have demonstrated their ability to tackle an array of problems across domains as diverse as healthcare, finance, and natural language processing. Deep learning, on the other hand, takes machine learning to a more intricate level by employing multilayer neural networks to simulate human brain processes. These neural networks hold the potential to not only distinguish nuanced changes within data but also to enhance decision-making within AI systems.

Deep Learning, a subdomain closely intertwined with Machine Learning, represents the next phase in the evolution of AI. What sets Deep Learning apart is its aspiration to simulate the intricacies of the human brain's neural processes during the decision-making phase. This is accomplished through the deployment of intricate multilayer neural networks, with multiple hidden layers that can uncover patterns and relationships within data, even when these are subtle and complex. As a result, Deep Learning has catalyzed breakthroughs in tasks such as image and speech recognition, visual object recognition, object detection or drug discovery (LeCun, Bengio, & Hinton, 2015). The essence of Deep Learning lies in its ability to replicate, to some extent, the human brain's neural processes. This is an ambitious endeavor, as the human brain is a marvel of biological engineering, characterized by its capacity to

process vast amounts of information and derive nuanced insights.

The success of deep learning is attributed in part to the availability of extensive labeled datasets and the exponential growth in computing power, particularly with the advent of Graphics Processing Units (GPUs). Deep learning models, notably Convolutional Neural Networks (CNNs) for image recognition and Recurrent Neural Networks (RNNs) for sequential data, have achieved groundbreaking performance in various domains. CNNs, for instance, have revolutionized computer vision by learning hierarchical features and patterns from images, enabling tasks like object detection, image classification, and segmentation. RNNs, with their ability to process sequential data while preserving contextual information, excel in tasks such as natural language processing, machine translation, and time-series analysis.

Deep learning networks, a subset of artificial neural networks (ANNs), represent a class of sophisticated models that leverage multiple layers of interconnected nodes to learn intricate representations of data. These networks have gained prominence owing to their ability to autonomously learn complex features from raw input data, enabling them to tackle challenging tasks in pattern recognition, natural language processing, and other domains. The distinguishing characteristic of deep learning networks lies in their depth, encompassing multiple hidden layers between the input and output layers, facilitating the extraction of hierarchical and abstract representations.

Convolutional Neural Networks (CNNs) stand as a cornerstone of deep learning networks, revolutionizing computer vision tasks. CNNs employ specialized layers such as convolutional layers, pooling layers, and fully connected layers to effectively process visual data. Convolutional layers apply a series of learned filters across the input image, extracting features like edges, textures, and shapes. Pooling layers subsequently down sample the extracted features, reducing computational complexity while retaining crucial information. These networks have demonstrated exceptional performance in image classification, object detection, image segmentation, and facial recognition, among other tasks. However, CNNs has disadvantages to be applied on small dataset causing overfitting (Yamashita, Nishio, Do, & Togashi, 2018).

Another class of deep learning networks, Recurrent Neural Networks (RNNs), excel in modeling sequential data by preserving temporal dependencies. RNNs maintain a memory of past inputs through recurrent connections, allowing them to capture sequential information in data sequences, such as sentences in natural language

processing or time-series data in finance or speech recognition. However, traditional RNNs encounter challenges in capturing long-term dependencies due to the vanishing or exploding gradient problem, which affects their ability to retain information over extended sequences.

To address the limitations of traditional RNNs, Long Short-Term Memory networks (LSTMs) (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Cho, et al., 2014) were introduced. LSTMs, featuring a more complex architecture with memory cells, input and forget gates, and output gates, excel in capturing and retaining long-range dependencies in sequential data. GRUs, a simplified variant of LSTMs with fewer parameters, offer similar capabilities in modeling sequential data, albeit with reduced computational overhead. Both LSTMs and GRUs have proven instrumental in tasks requiring the understanding of context and relationships in sequential data, such as language translation, sentiment analysis, and speech recognition.

Deep learning networks continue to evolve with innovations like Transformer architectures, attention mechanisms, and self-supervised learning paradigms. These advancements aim to enhance the capabilities of deep learning models in understanding complex relationships in data, handling diverse modalities, and improving generalization to previously unseen examples. As research in deep learning progresses, the versatility and adaptability of these networks hold promise for solving intricate real-world problems across various domains, driving advancements in technology and reshaping industries with their transformative capabilities.

The synergy between AI, Machine Learning, and Deep Learning continues to drive innovation, creating a profound impact on how we live and work. These technologies are shaping our present and defining our future, with the potential for ever more sophisticated applications across numerous domains.

Deep Learning stands out for its ability to understand subtle details or changes in data using complex layers of interconnected neurons. This intrinsic complexity, while empowering the discernment of subtle patterns and abstract representations, poses challenges during the model's developmental phases. Particularly, the configuration and calibration of hyperparameters demand meticulous attention before initiating the data training phase. Determining parameters such as learning rates, network architecture, and regularization techniques becomes a meticulous and time-consuming endeavor, impacting the model's convergence, robustness, and overall

performance.

Moreover, the interpretability in Deep Learning is an issue due to its complexity with intricate layers and interconnectedness of neurons in deep neural networks (DNNs). The 'black box' nature of these models raises concerns regarding transparency, making it arduous to decipher the inner workings and rationale behind the model's predictions. This lack of interpretability prevents the trustworthiness and adoption of Deep Learning models, particularly in domains where comprehensible decision-making processes are imperative, such as healthcare and finance. Efforts to enhance interpretability involve developing methods that shed light on the model's decision-making process, enabling stakeholders to comprehend and trust the outcomes, thereby augmenting the model's utility and reliability.

Furthermore, the complexity deep learning introduces to the process is noteworthy. While it excels at discerning subtle variations and nuanced changes within data through its intricate web of interconnected neural networks, it simultaneously exacerbates the initial stages of model development. Specifically, the definition and determination of hyperparameters become arduous tasks before data training can commence.

The challenges encountered in deep learning, such as the necessity of extensive datasets and the intricate nature of hyperparameter tuning, are reflective of the broader challenges faced in the AI landscape. As AI systems continue to evolve, they demand more robust data processing capabilities and refined model architectures. Machine learning and deep learning, as integral components of AI, must cohesively address these challenges to unlock the full potential of artificial intelligence in various technological domains. One of the fundamental challenges in Deep Learning is the necessity for vast amounts of high-quality data. DNNs thrive on large datasets for training, yet collecting and curating such datasets can be a daunting task. Additionally, the quality and diversity of the data are paramount. Biased or incomplete datasets can lead to skewed model outcomes, affecting the fairness and reliability of the model's predictions. Deep Learning models are computationally demanding, requiring powerful hardware infrastructure for training and inference. This poses challenges in terms of access and affordability. Training DNNs on standard personal computers can be time-consuming or even impractical. Consequently, access to high-performance computing clusters or specialized hardware is often necessary, which can limit the accessibility of this technology to smaller organizations or individuals. DNNs, with

their multiple hidden layers, are capable of modeling intricate relationships in data. However, this complexity can also lead to overfitting, where the model performs exceptionally well on the training data but fails to generalize effectively to unseen data. Managing model complexity through techniques like dropout and regularization is crucial to prevent overfitting. Determining the optimal hyperparameters for deep learning models can be challenging. Hyperparameters, such as learning rates, batch sizes, and the number of layers, significantly impact model performance. Finding the right combination of hyperparameters often requires extensive experimentation and can be a time-consuming process. Deep Learning models are often described as "black boxes" due to their complexity. Understanding how these models arrive at specific decisions can be challenging, especially in fields where interpretability is crucial, such as healthcare and finance. Developing techniques to make deep learning models more interpretable is an ongoing challenge. While deep learning models can achieve exceptional performance in specific domains, transferring knowledge from one domain to another remains challenging. Generalization across diverse datasets and tasks is an active area of research, as it holds the potential to make deep learning more versatile and applicable across a broader range of domains. The issue of ethics and fairness is also another challenge in Deep Learning (Du, Yang, Zou, & Hu, 2020). Biases present in training data can perpetuate discrimination in model predictions, resulting in unfair outcomes. Addressing these ethical concerns and developing methods for bias mitigation is an ongoing challenge for the deep learning community. Additionally, large deep learning models can be unmanageable to deploy in real-world applications with limited computational resources, such as edge devices or IoT devices. Therefore, model compression techniques that reduce model size without significant loss of performance are vital for practical deployment. The challenges in Deep Learning are multi-faceted, ranging from data-related issues to ethical concerns and the need for ongoing model refinement. Addressing these challenges is essential to harness the full potential of DNNs and advance the state of the art in AI applications.

Deep ensembling is a powerful technique in the field of machine learning, particularly in deep learning, where multiple neural networks are combined to enhance predictive accuracy and model robustness (Lakshminarayanan, Pritzel, & Blundell, 2017). The process involves training and combining multiple DNNs, each with different initializations, architectures, or subsets of the data. This technique helps to improve generalization, stability, accuracy while reducing the variance in model

predictions.

Pruning plays a pivotal role in deep ensembling, where multiple neural networks are combined to enhance predictive accuracy and model robustness. Deep ensembles often involve a multitude of neural networks, and not all of them may be essential for the ensemble's performance. By pruning these unnecessary elements, the ensemble becomes more computationally efficient, leading to reduced memory usage, faster training, and improved resource utilization. Ensemble pruning enhances the generalization capabilities of deep ensembles. Deep ensembles are known for their ability to reduce overfitting, and pruning further addresses this issue by removing models or components that might overfit to the training data. The result is an ensemble that is more robust and reliable when faced with unseen data, a critical feature for real-world applications. Pruning not only reduces the complexity of ensembles but also results in model compression. Smaller models require less storage space, which is a valuable consideration when deploying models on devices with limited memory. However, discovering a compact set of classifiers that can match the performance of a boosted ensemble is a computationally challenging task, known as an NP-hard problem (Tamon & Xiang, 2000). As an alternative, search-based methods come into play for assembling an ensemble. These methods explore the ensemble space, assessing the performance of different subsets of classifiers (Sagi & Rokach, 2018). It's worth noting that deep learning itself can be viewed as a form of ensemble learning, as it comprises multi-layer artificial neural networks.

In response to these challenges, the research community has delved into the exploration of various pruning techniques that aim to streamline deep learning models. Notable contributions in this area include proposals for reducing the parameters of convolutional networks (Han, Mao, & Dally, 2016; Han, Pool, Tran, & Dally, 2015; Hanson & Pratt, 1989; LeCun, Denker, & Solla, Optimal Brain Damage, 1990; Ström, 1997). These pruning methods have been introduced in an attempt to alleviate the computational burden and simplify the training process. However, they have often yielded minimal performance improvements, primarily concentrating on the reduction of network weights. While these pruning techniques have made notable efforts to optimize deep learning models, they have not been the panacea for all challenges faced in this field. The common problem of deep learning and pruning algorithms is the decision of the pruning percentage with the heuristic approach at the pruning stage, making the success percentage of the deep learning algorithm dependent on the

pruning parameter. On the other hand, the optimization models proposed with zero-norm (ℓ_0) penalty to ensure sparsity ignore the diversity of the layers as they only take into account the percentage of success. The combination of the layers that are close to each other does not increase the percentage of accuracy. Sparse problem minimization and techniques for low-order approximation have found widespread applications in various domains, including computer vision, machine learning, and telecommunications, among others (Quach, Duong, Luu, & Bui, 2017). In such problems, incorporating the concept of sparsity, often represented by the zero-norm (ℓ_0), results in non-convex objective functions within optimization tasks. Consequently, the optimization community has introduced various relaxation techniques to address non-convex objective functions. Specifically, for a class of non-convex quadratic constrained problems, SDP (Semidefinite Programming) relaxation techniques have been developed based on the Difference of Convex (DC) decomposition strategy, aimed at reformulating the problem into the difference of two convex functions (Zheng, Sun, & Li, 2011; Özöğür-Akyüz, Otar, & Karadayı Ataş, 2020). However, it's important to note that SDP algorithms tend to be slow and resource-intensive when applied to high-dimensional problems. As a solution, SDP algorithms have been transformed into quadratic conic problems to improve efficiency.

In summary, artificial intelligence, machine learning, and deep learning are interconnected domains that hold immense promise for our contemporary society. As technological advancements continue, the challenges they face necessitate a multidisciplinary approach to further their development. While pruning techniques have been introduced to address certain complexities, the interplay between these fields remains a key area of exploration, enabling AI to reach new heights in its applications and capabilities.

This thesis introduces an innovative optimization model designed to streamline the ensemble of Convolutional Neural Networks (CNNs) using a Second-Order Cone Programming (SOCP) framework. The primary objective of this novel optimization model is to identify an optimal subset of models within the CNN ensemble. This selection process aims to strike a delicate balance between accuracy and diversity while concurrently diminishing the overall quantity of models and associated computational intricacies. Furthermore, this model has been adapted to function as a feature selection algorithm, denominated as FSOCP (Feature Selection via Second-

Order Cone Programming). This adaptation serves the purpose of identifying the most pertinent features, thus enhancing the overall classification accuracy. The contributions of this thesis can be listed as follows:

- A distinctive contribution this proposed model offers is its direct acquisition of the pruning percentage parameter through the proposed second-order conical optimization model. This parameter, specifically tailored to the demands of deep learning research, will be determined directly by this model.
- Furthermore, a significant novel contribution lies in providing a remedy to the lack of consideration for diversity criteria in existing literature, where deep learning network pruning has traditionally been based solely on accuracy performance. In the proposed optimization model, the objective function concurrently addresses both accuracy and diversity criteria by streamlining the ensemble of CNNs.
- Considering that SOCP models consistently yield more effective and expeditious outcomes compared to alternative sparse optimization models, the application of the proposed SOCP model will lead to a substantial reduction in the quantity of models within the CNN ensemble.
- The thesis also assesses the performance outcomes of diverse feature selection methods from existing literature on a range of datasets and compares them with the novel feature selection approach, FSOCP. The results indicate that FSOCP serves as a viable alternative, demonstrating favorable outcomes in feature selection when applied.
- The FSOCP feature selection model, introduced in this thesis, exhibits superior efficiency when compared to the alternative methods. This efficiency is evident in both the training time of the algorithm and the overall processing time. This advantage becomes particularly pronounced when compared to methods found in the existing literature, as they employ a brute force approach to identifying feature subsets. In such methods, the search space expands exponentially as the number of features grows, consequently leading to heightened computational demands.
- Moreover, the proposed approach stands out as an optimization-driven model, providing a clear and transparent mathematical rationale. This sets it apart from techniques that depend on thorough examination of all potential scenarios, as seen in brute-force methods. This particular attribute contributes to an improved understanding of the algorithm's internal mechanisms.

This thesis is organized as follows; literature review will be given in Chapter 2

and it will be separated in two parts; Section 2.1 will discuss studies in the literature on deep learning, neural networks and pruning in general while Section 2.2 will be on sparse optimization models and power of sparse optimization problems. Chapter 3 will mention the proposed methods how these methods are developed according to mathematical proofs and definitions with explaining the steps for generating an ensemble of CNNs. This chapter will be separated into three parts; Section 3.1 introduces the loss function and how it emerges by following sparse optimization proofs and entropy and in Section 3.2 the proposed loss function will be reformulated as second-order cone programming model. The Section 0 explains how the proposed model is implemented as an embedded feature selection algorithm and other feature selection algorithms in the literature will be mentioned. In Chapter 4, results of the proposed technique will be shared in two subsection where the Section 4.1 will give the results of the pruning of CNNs with SOCP and Section 4.2 will give results of the proposed feature selection algorithm and its performance compared to the other feature selection algorithms in the literature. Lastly, Chapter 5 will be related to discussions of the proposed model which is separated two subsection where the Section 5.1 mentions the proposed pruning method of CNNs and its importance by giving future outlook and the Section 5.2 explains the same for the proposed feature selection method.

Chapter 2

Literature Review

The collaboration between Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) stands as the cornerstone of contemporary technological progress, reshaping various industries and redefining methods for problem-solving. Recognizing their importance and the complex connections between these areas reveals the considerable potential they jointly possess for transformation.

AI encompasses the development of systems that emulate human intelligence, aiming to perform tasks like reasoning, learning, problem-solving, and perception. AI's significance lies in its potential to automate processes, augment decision-making, personalize experiences, and address complex problems across diverse fields, from healthcare and finance to transportation and entertainment. Machine Learning is a core component of AI, providing the learning mechanisms necessary for AI systems to adapt, improve, and make decisions based on data. AI systems leverage ML techniques to process and learn from data, enabling them to perform tasks such as recognizing patterns, making predictions, and optimizing processes. ML drives advancements in predictive modeling, recommendation systems, fraud detection, language translation, and personalized user experiences, fostering innovation and efficiency in various industries.

Deep Learning is a subset of ML involving neural networks with multiple layers that learn hierarchical representations of data where it has revolutionized computer vision, natural language processing, speech recognition, and robotics, enabling breakthroughs in image classification, language translation, and autonomous systems. DL techniques, such as convolutional and recurrent neural networks, constitute a significant portion of ML advancements, contributing to image recognition, natural language understanding, and complex pattern recognition tasks.

In an era characterized by the rapid advancement of deep learning and its widespread utilization, Convolutional Neural Networks (CNNs) have emerged as a cornerstone in the realm of artificial intelligence. These neural architectures have displayed remarkable proficiency across a spectrum of applications, encompassing tasks from image classification to natural language processing. Yet, as CNNs continue to progress, they give rise to substantial computational and memory challenges, especially in complex, high-dimensional scenarios. To confront these challenges and

enhance the efficiency and comprehensibility of CNNs, the research domain of 'pruning' has gained prominence. Pruning involves the systematic removal of non-essential model components while preserving performance integrity. Moreover, the integration of sparse optimization models has paved the way for achieving sparsity, thereby elevating computational efficiency and reducing model redundancy. This thesis embarks on an exploration of the utilization of sparse optimization models for the purpose of pruning CNNs, with a specific emphasis on managing the intricate balance between model intricacy and predictive accuracy within the realm of deep learning where the proposed optimization model is also used as a feature selection algorithm. Subsequent chapters delve into the theoretical foundations, practical methodologies, and empirical validations of this approach, culminating in a substantial contribution to the field of deep learning.

2.1 Optimizing Neural Networks: Deep Learning Network Pruning, Crafting Deep Ensembles, and Fine-Tuning Ensemble Pruning for Efficient AI Models

Alongside technological advancements, artificial intelligence holds a significant place in contemporary society. Machine learning, one of the key subfields of artificial intelligence, aims to develop predictive models for data using various algorithms. Deep learning, a subdomain closely related to machine learning, also engages in data-driven learning. However, it distinguishes itself from machine learning by utilizing multilayer neural networks that emulate the neural processes in the human brain during decision-making.

In recent years, deep learning has shown considerable promise in various technological domains. Nevertheless, it faces certain challenges that necessitate further development. Primarily, there is a need to train extensive datasets. Training these vast datasets requires a substantial amount of processing power. While deep learning can distinguish subtle changes and fine differences in data through interconnected neural networks, defining hyperparameters before data training proves to be a complex task. As a result, the literature has proposed various pruning methods to reduce the parameters of convolutional (Han, Mao, & Dally, 2016; Han, Pool, Tran, & Dally, 2015; Hanson & Pratt, 1989; LeCun, Denker, & Solla, Optimal Brain Damage, 1990; Ström, 1997). However, these pruning methods have focused on reducing the number of network weights, leading to minimal improvements in performance. Alternatively, a study introduced the Runtime Neural Pruning (RNP) method (Lin, Rao, Lu, & Zhou, 2017). This method dynamically prunes artificial neural networks during operation.

To achieve this, each convolutional layer's pruning is modeled as a Markov decision process (MDP), and each agent is trained using reinforcement learning to optimize the pruning process (Lin, Rao, Lu, & Zhou, 2017). Runtime neural pruning is a dynamic approach within neural network optimization that involves the selective removal or deactivation of neurons or connections during the inference or runtime phase, rather than during the training phase. This technique aims to enhance the efficiency and computational speed of neural networks by reducing their size or complexity while preserving performance. Unlike conventional pruning methods applied during training, which remove connections based on predefined criteria before deployment, runtime pruning occurs dynamically as the network processes data, adapting to input characteristics or network states. The concept of runtime neural pruning leverages various criteria to identify and eliminate redundant or less influential components of a neural network during its operational phase. This adaptability allows the model to adjust its architecture in response to input patterns or activation levels, optimizing computational resources without compromising accuracy. Techniques such as activation-based pruning capitalize on the observation that certain neurons or connections remain dormant or contribute minimally to the network's output for specific inputs. By deactivating or pruning these less active components during runtime, computational overhead can be significantly reduced, enhancing the model's efficiency, particularly in resource-constrained environments like edge devices or real-time applications. Runtime neural pruning holds promise for improving the efficiency and deployment viability of neural networks by tailoring their architecture dynamically to the task at hand. This approach minimizes computational resources by adapting the network's structure in real-time, effectively reducing memory and processing requirements without sacrificing accuracy. However, implementing runtime pruning introduces challenges in designing efficient algorithms capable of dynamically identifying and modifying network components during inference without compromising the model's predictive capabilities. Nonetheless, the potential for runtime neural pruning to optimize neural network efficiency during deployment underscores its significance in advancing the practicality and scalability of deep learning models across various domains. The proposed (Lin, Rao, Lu, & Zhou, 2017) RNP method's flowchart is depicted in Figure 1:

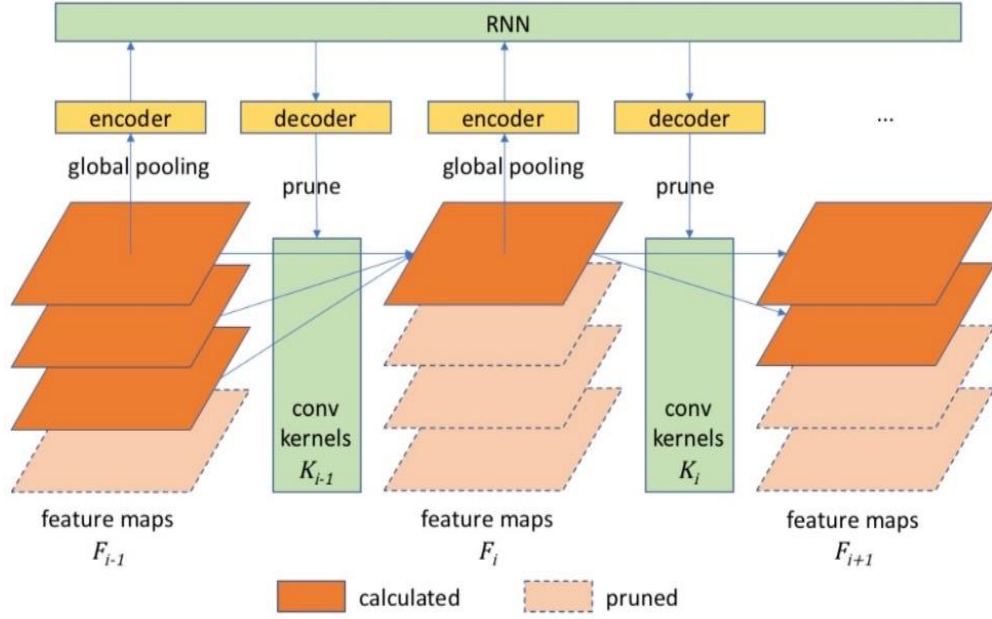


Figure 1 The general structure of RNP (Lin, Rao, Lu, & Zhou, 2017).

In a recent study aimed at enhancing the performance of deep learning, powerful artificial neural networks were intentionally weakened, and one of the diversity techniques, bagging, was applied. The study demonstrated that weakening artificial neural networks and combining this with a diversity technique could yield performance benefits. Notably, it showed significant improvements, particularly in binary classification tasks (Alvear-Sandoval & Figueiras-Vidal, 2015).

The primary objective of pruning is to remove unnecessary parameters, thereby improving the training process. In pursuit of this objective, a pruning method dependent on functionality vectors, known as the Distinctiveness Reduction Technique, was employed for optimizing the elimination of hidden neurons in fully connected neural networks (Zhou Z., 2018; Thakur, 2018). This method was also applied to convolutional networks, where it was observed that in dense feature sets, feature extraction becomes complex and accuracy decreases (Thakur, 2018).

In recent years, a study introduced a genetic algorithm-based channel pruning method for very deep Convolutional Neural Networks (CNNs). Channel selection is an exponentially combinatorial optimization problem within the solution space. To expedite the selection process, the proposed method modeled the channel selection problem as a search problem and solved it using a genetic algorithm. The two-stage method was devised to enhance the efficiency of the genetic algorithm's processing

time. Initially, it pruned each layer of the pre-trained CNN model based on sensitivities. Subsequently, this pruned model was fine-tuned according to the knowledge distillation framework. These adjustments significantly reduced model redundancy without sacrificing accuracy (Hu, Sun, Li, Wang, & Gu, 2018).

In the literature, an efficient and traceable pruning method known as Net-Trim was introduced to simplify trained neural networks. In this method, a convex program was defined for layer-wise pruning of trained networks without disturbing internal responses (Aghasi, Abdi, Nguyen, & Romberg, 2017). The Net-Trim method resembles Lasso in its approach to finding sparsity in weight sets; both methods employ norm-1 (ℓ_1) for sparsity evaluation (Aghasi, Abdi, Nguyen, & Romberg, 2017). Since the choice of regularization is crucial for pruning performance, ℓ_1 and norm-2 (ℓ_2) were separately evaluated. However, it was observed that ℓ_1 penalized non-zero parameters, resulting in a more significant number of parameters approaching zero. While this selection yielded better accuracy before retraining the network, it was noted that the remaining connections were not as effective as ℓ_2 (Han, Mao, & Dally, 2016). The graph in Figure 2 below illustrates the relationship between the pruning percentage and the error percentage for this method, which is regularized using different norms:

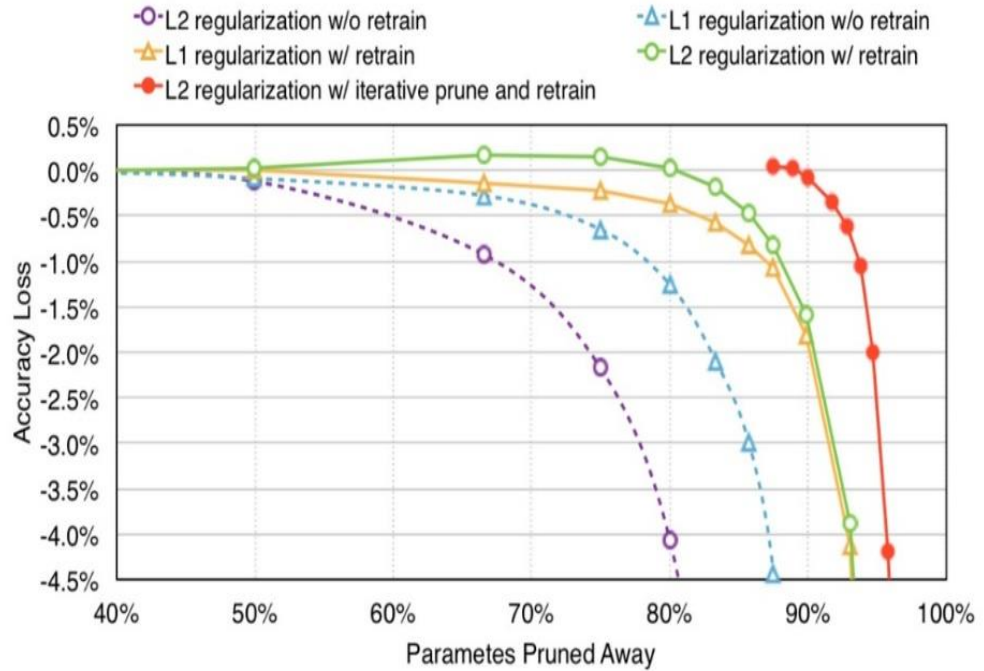


Figure 2 Performance results due to selection of different norms (Han, Mao, & Dally, 2016).

Various mathematical optimization methods have been proposed to reduce the computational overhead of DNNs during training and meaningful inference stages. Optimized results are crucial for minimizing power consumption and latency. Training DNNs requires labeled datasets, employing the back-propagation algorithm to make small weight adjustments in network connections using gradient descent. Consequently, research in the literature has shifted the focus from optimizing the training phase to optimizing the stage where meaningful results are obtained (Green, Vineyard, & Koç, 2018).

DNNs are composed of different layers, with each layer designed to address a specific challenge or weakness. Each classification problem can be optimally addressed by a different architectural arrangement or combination of layers. Convolutional and fully connected layers are the layers in DNNs where the majority of meaningful processing takes place and thus, they have been the focus of optimization efforts (Green, Vineyard, & Koç, 2018). Mathematical optimization approaches to DNNs can be categorized into five main groups in the literature: pruning, quantization, weight sharing and compression, model distillation, and filter decomposition. Except for filter decomposition, all other methods are applied to both convolutional and fully connected layers. Filter decomposition is exclusively used in convolutional layers and aims to modify these layers to reduce the number of weights along with latency. Pruning reduces the number of weights, required computational operations, and storage demands. Quantization is applied to both of these layers and reduces the number of bits in the precision of inputs to neural networks, thus lowering memory requirements. Weight sharing and compression methods enforce sharing weights to create space for memory. Model distillation reduces the weight count and latency by training a smaller network to mimic a larger one. However, it has been observed that all these optimization models lead to a compromise in accuracy (Green, Vineyard, & Koç, 2018).

To handle the high computational burdens in deep convolutional networks, the identification and elimination of insignificant filtering operations have been proposed. However, the negative aspect of these approaches lies in the layer-by-layer filtering and the necessity for fine-tuning iterations. Therefore, in a study in the literature, an efficient reduction of filtering operations was proposed. This method, based on Partial Least Squares (PLS) and Variable Importance in Projection (VIP), identifies the importance of each filter without compromising accuracy, thus eliminating the

insignificant filters effectively (Jordao, Kloss, Yamada, & Schwartz, 2018). This proposed method is illustrated in Figure 3 below:

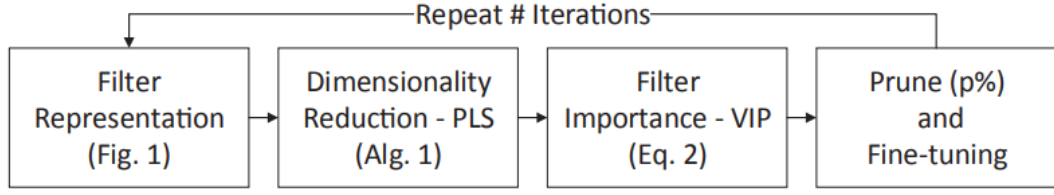


Figure 3 The steps for pruning convolutional filters in deep networks (Jordao, Kloss, Yamada, & Schwartz, 2018).

The proliferation of devices connected to the internet has introduced the concept of the Internet of Things (IoT). Machine learning algorithms have been proposed to address the dynamic changes in this concept, offering promising solutions (Vermesan & Friess, 2014). DNNs have been noted to introduce delays and increased energy consumption in IoT devices due to their complexity and excessive parameters (Ardakani, Condo, & Gross, 2017). While pruning methods in the literature have mainly focused on reducing the number of parameters and have successfully reduced power consumption in fully connected layers, they have not shown noticeable speedup in convolutional layers (Wen W. , Wu, Wang, Chen, & Li, 2016). To address this, an activation pruning method was developed to reduce the number of activations in each convolutional layer, which affects system delays, and the performance of the pruned network was compared to the unpruned network. The pruned network outperformed the unpruned network in terms of the misclassification rate (Ardakani, Condo, & Gross, 2017). Similarly, the researchers proposed a method to reduce the number of kernels in convolutional networks, achieving a reduction in accuracy by up to 90%. Randomly deleted connections showed significant performance, and based on this idea, the proposed architecture aims to accelerate fully connected and convolutional layers with a single computational core, which can reduce the budget on mobile devices (Ardakani, Condo, & Gross, 2017).

In practice, deep learning algorithms are believed to underperform in real-world scenarios due to their functional complexity and tight interaction with memory. Although network pruning seems like a viable option, it can lead to irregular network connections and may not conform to parallel computation. Structured sparsity methods have been proposed for convolutional networks, including channel-wise, kernel-wise,

and intra-kernel strided sparsity, to address these issues (Anwar, Hwang, & Sung, 2017). The method proposed using particle filtering approach for determining the importance weight of each component based on the misclassification rate and observed that intra-kernel strided sparsity reduced computational overhead (Anwar, Hwang, & Sung, 2017).

Transfer learning is a machine learning technique that leverages knowledge gained from solving one problem and applies it to a different but related task. This approach circumvents the need to train a model from scratch for every new task, instead, it transfers knowledge and representations learned from previous tasks to accelerate learning or enhance performance on a new task. One of the key advantages of transfer learning is its ability to expedite model training, particularly in scenarios where labeled training data for the target task is limited. By utilizing pre-trained models that have learned from vast amounts of data in related domains, transfer learning can significantly reduce the need for extensive labeled data, thereby expediting the training process and potentially improving the model's generalization to new tasks.

Transfer learning often leads to enhanced performance and better generalization (Wang & Chen, 2022). By initializing a model with weights and features learned from pre-trained models trained on large datasets, it provides a starting point that encapsulates generic patterns and representations. This initialization tends to boost the performance of the model on the target task, especially when the source and target tasks share similarities in data distribution or features. The transferred knowledge can help the model learn relevant features more effectively and adapt to the nuances of the new task, resulting in improved accuracy and faster convergence during training (Wan, Xiong, Li, Zhu, & Huan, 2019). However, transfer learning is not without its drawbacks. One notable disadvantage is the risk of negative transfer, where the knowledge transferred from the source task may not be entirely beneficial or relevant to the target task. If the source and target tasks differ significantly in their underlying data distributions or task characteristics, the transferred knowledge might hinder rather than improve the model's performance. Negative transfer can occur if the learned features or representations from the source task conflict with or misguide the learning process for the target task, leading to suboptimal performance.

Pruning convolutional neural networks in the context of transfer learning has been investigated. Trimming one network component with the First-Order Taylor

Approximation to reduce training cost was proposed. This method eliminates the need for sensitivity analysis for each layer and employs the ℓ_0 -norm. The study found that intuitive approaches could successfully prune the least important feature maps step by step in convolutional neural networks (Molchanov, Tyree, Karras, Aila, & Kautz, 2016).

In the literature, a pruning method for dividing large datasets into blocks for efficient processing has been proposed (Sheng, Li, Wang, Deng, & Zhao, 2015). To achieve pruning learning, a model containing both hard and soft constraints, accommodating the min-k inequality, has been developed. The method provided significant improvements in training time (Sheng, Li, Wang, Deng, & Zhao, 2015).

Studies have been conducted on pruning the weights and connections of Long Short-Term Memory (LSTM) networks, which are a type of artificial recurrent neural network (RNN). One of these studies is based on the idea of eliminating connections with smaller weights compared to others among the neurons, followed by retraining the model and achieving a sparse weight matrix (Tang & Han, 2015). Accordingly, a pruning model based on the standard deviation of the weight distribution in a pre-trained model is developed and a threshold was determined to decide the pruning rate (Tang & Han, 2015).

In structured prediction tasks with user interaction in gradual architectures, a multiple-choice learning (MCL) algorithm has been proposed where a loss function has been suggested to construct the problem effectively, and a max-margin formulation has been proposed for training M-tuples (Guzman-Rivera, Batra, & Kohli, 2012).

In the literature, a stochastic gradient descent (SGD) approach for pruning connections during the learning phase of the network has been developed within the TensorFlow framework (Abadi, et al., 2016). For each selected layer to be pruned, a binary mask variable of the same size and shape as the layer's tensor weight has been added. Operations were performed to reduce the smallest weight values to zero to achieve the desired sparsity levels. The accuracy of the model during the learning phase is related to pruning unnecessary connections in the initial phase and the number of pruned weights. SGD generally decreases the learning rate during training, which makes it difficult for pruning to recover from the loss caused by forcing weights to zero in subsequent training steps. However, pruning with a very high learning rate can lead to poor pruning weights when a good solution has not yet been found, making the choice of pruning tactic closely related to the learning rate (Zhu & Gupta, 2018).

Research in the field of deep neural network pruning can be categorized into different methodologies. These encompass techniques such as Biased Weight Decay (Hanson & Pratt, 1989), Optimal Brain Damage (Cun, Denker, & Solla, 1990), and Optimal Brain Surgeon (Hassibi & Stork, 1992). Alternative approaches focus on identifying unnecessary parameters signified by connections within network architectures, eliminating these redundancies for pruning (Srinivas & Babu, 2015; Han, Mao, & Dally, 2016). Additionally, a separate study introduced a layer-specific pruning method for DNNs (Dong, Chen, & Pan, 2017). This method independently pruned the parameters of each layer based on the second-order derivatives of a layer-specific loss function concerning the relevant parameters. Importantly, it was demonstrated that the reduction in prediction performance after pruning was constrained to a linear combination of the restructured errors emerging in each layer (Dong, Chen, & Pan, 2017).

In recent times, there has been a growing interest in achieving compact network designs with minimal performance impact through pruning, particularly in the context of Convolutional Neural Networks (CNNs) (Wen W. , Wu, Wang, Chen, & Li, 2016; Zhou, Wang, Li, & Xu, 2016). It's important to note that pruning techniques may differ in terms of their approach to network structure, scoring mechanisms, pruning schedules, and fine-tuning criteria (Blalock, Ortiz, Frankle, & Gutttag, 2020). Recently, there has been significant interest in obtaining compact designs with minimal impact on performance through pruning (Wen W. , Wu, Wang, Chen, & Li, 2016; Zhou, Wang, Li, & Xu, 2016), especially when using Convolutional Neural Networks (CNNs). It is worth noting that pruning techniques may vary in terms of their approach to structure, scoring, scheduling, and fine-tuning selection (Blalock, Ortiz, Frankle, & Gutttag, 2020).

Some pruning methods concentrate on individual parameters, causing sparsity through unstructured pruning. While they reduce the architecture's memory footprint, they don't necessarily offer speed advantages. In contrast, other techniques focus on groups, treating neurons, filters, or layers as distinct pruning targets, which has been noted to improve hardware utilization (Li, Monroe, & Jurafsky, 2016; He, Zhang, & Sun, 2017). Parameters can be evaluated based on their absolute value, relevance to training metrics, activation, and contribution to gradients. Some approaches assess parameters locally within closely connected groups, rather than considering the entire architectural entity (Lee, et al., 2019; Frankle & Carbin, 2019). The literature reports

that pruning algorithms exhibit varying speeds on a per-step basis, with some employing iterative pruning by targeting specific architectural subsets (Liu, Xin, Mu, & Zhu, 2019; Han, Mao, & Dally, 2016). Certain methods adjust the pruning rate following defined rules (Gale, Elsen, & Hooker, 2019). After applying pruning algorithms, some techniques persist with the same training weights, while others retrain the reduced architecture up to a specific point or reset it to the initial state and initiate training anew (Frankle & Carbin, 2019; Liu, Xin, Mu, & Zhu, 2019). Figure 4 below illustrates the state of artificial neural networks before and after pruning.

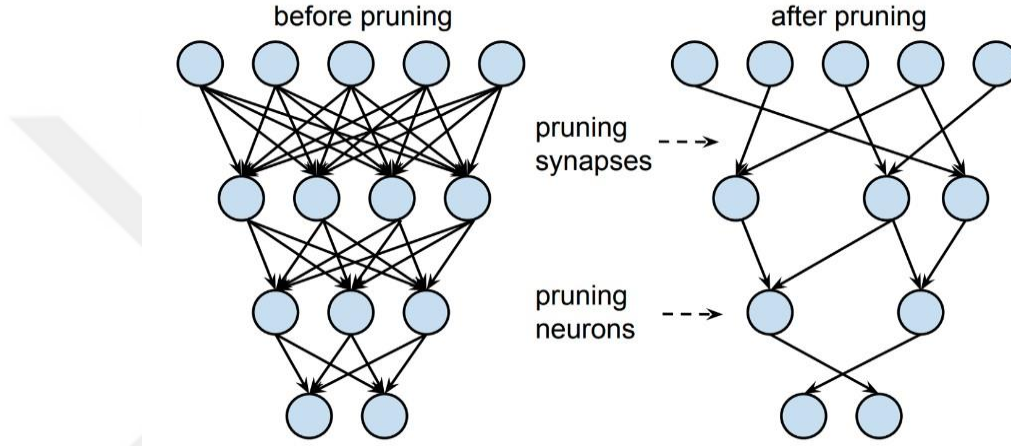


Figure 4 The state of neural networks before and after pruning (Liu, Xin, Mu, & Zhu, 2019).

A recurrent issue in recent years within deep learning and pruning algorithms is the reliance on heuristic approaches for determining the pruning percentage, which in turn links the success of deep learning algorithms to this pruning parameter. Conversely, optimization models that leverage zero-norm (ℓ_0) penalties to induce sparsity often overlook the diversity among layers and merely focus on success rates. Merging neighboring layers, under these models, does not necessarily lead to an increase in accuracy percentages.

Deep ensemble techniques enhance the reliability of DNNs by combining multiple diverse models into a more robust model (Gale, Elsen, & Hooker, 2019). These ensemble models have been demonstrated to yield different predictions for the same input, thereby increasing diversity and resulting in more accurate probability estimates (Garipov, Izmailov, Podoprikin, Vetrov, & Wilson, 2018; Lakshminarayanan, Pritzel, & Blundell, Simple and scalable predictive uncertainty

estimation using deep ensembles, 2017). In a recent study, an ensemble approach was employed to augment complexity by training two EfficientNet-b0 models end-to-end using bagging. An adaptive ensemble technique was implemented by fine-tuning within a trainable combination layer, surpassing the performance of various other studies on well-known datasets such as CIFAR-10 and CIFAR-100 (Antonio, Moroni, & Martinelli, 2023). Leveraging pre-trained models has proven to enhance efficiency and streamline hyperparameter tuning, leading to improved performance on these datasets through transfer learning and pre-training (Kolesnikov, et al., 2020) (Sun, Shrivastava, Singh, & Gupta, 2017). While researchers have accelerated Convolutional Neural Networks (CNNs), they have observed reduced performance metrics on CIFAR-100. One such technique is the FATNET algorithm, which employs high-resolution kernels for classification while reducing the number of trainable parameters based on the Fourier transform (Ibadulla, Chen, & Reyes-Aldasoro, 2023).

Ensemble pruning methods serve as strategies to enhance the efficiency and performance of ensemble learning models by refining or simplifying the ensemble structure. Ensembles typically consist of multiple base models or learners whose collective predictions are aggregated to generate a final prediction. Pruning techniques aim to optimize the ensemble by either reducing its complexity, minimizing redundancy among base models, or eliminating weaker or less informative models. One prevalent method in ensemble pruning involves model selection, where models demonstrating weaker performance or contributing less to the ensemble's overall accuracy are pruned or excluded from the final ensemble. This process helps prevent overfitting and enhances the generalization ability of the ensemble by retaining only the most pertinent and diverse models. The task of identifying a small subset of classifiers that can perform on par with a boosted ensemble is a computationally complex problem, as established in prior research (Tamon & Xiang, 2000). Search-based approaches offer an alternative by exploring the ensemble space and assessing the performance of different classifier subsets (Sagi & Rokach, 2018). Feature-based ensemble pruning methods focus on selecting or weighting features that significantly contribute to the ensemble's predictive power. These methods identify and retain the most informative features, eliminating redundant or less relevant ones, thereby streamlining the ensemble's feature space. By reducing the number of features used by the ensemble, feature-based pruning techniques mitigate the risk of overfitting, improve computational efficiency, and enhance the interpretability of the ensemble.

Deep learning can be conceptualized as a multi-layer artificial neural network, which essentially operates as an ensemble learning method. The overall success rate in ensemble learning is linked to both the average accuracy of the ensemble and the diversity of individual learners within the ensemble. However, accuracy and diversity exhibit trade-offs, implying that an increase in accuracy within the ensemble may lead to reduced diversity, indicating the convergence of similar layers. It has been demonstrated that the pruning challenge can be reformulated as a quadratic integer problem for classification, aiming to identify a subset that optimizes the accuracy-diversity trade-off using semi-definite programming techniques (Zhang, Burer, & Street, 2006). Building upon this research, several optimization models have been proposed to leverage the quadratic nature inherent in pruning problems (Güldoğuş, Abdullah, Ali, & Akyüz, 2023).

2.2 Unlocking Efficiency: Sparse Optimization Models and the Power of Second-Order Cone Programming in Pruning AI Models

Sparse optimization techniques play a pivotal role in various fields, offering solutions characterized by a limited number of non-zero elements. They enable simplicity, interpretability, and computational efficiency in solving problems across domains like signal processing, machine learning, and statistics. Understanding and leveraging sparse optimization techniques are crucial for addressing high-dimensional data problems and enhancing model interpretability while maintaining predictive performance.

Sparse optimization aims to find solutions where the variables or parameters are mostly zero, with only a few elements being non-zero. This concept is valuable in scenarios where simplicity or sparsity in the solution is preferred, reducing complexity and improving interpretability. In many optimization problems, adding a regularization term that promotes sparsity is crucial. Commonly used norms include the ℓ_1 -norm (Lasso) or the ℓ_0 -norm (which counts the number of non-zero elements).

Norms serve as crucial mathematical tools, providing a framework to measure distances, define convergence, and quantify magnitudes in vector spaces and beyond. Their properties and applications span various disciplines, from geometry and functional analysis to machine learning and optimization, influencing how distances and magnitudes are understood and utilized in mathematical contexts. A norm is a function that assigns a non-negative value to each vector in a vector space and it is denoted as $\|\cdot\|$ or $\|\cdot\|_p$. The widely known norms are;

- Euclidean Norm (ℓ_2 -norm)

The Euclidean norm, often referred to as the 2-norm or ℓ_2 -norm, is a fundamental concept in mathematics and vector spaces. It defines the length or magnitude of a vector in Euclidean space, providing a measure of its distance from the origin.

Geometrically, the Euclidean norm represents the length of the vector, considering its components as coordinates in space. In two-dimensional space, it corresponds to the length of the vector from the origin to a specific point in the Cartesian plane. In three dimensions, it extends this concept to measure the magnitude of vectors in 3D space, and it generalizes naturally to higher dimensions.

The Euclidean norm finds extensive applications in various fields, including physics, engineering, data analysis, and optimization. It serves as a crucial tool in calculating distances between points, forming the basis for distance metrics in machine learning algorithms. Additionally, it plays a significant role in defining the convergence criteria for iterative algorithms and optimization techniques by measuring the change or difference between successive iterations.

- L1-norm (ℓ_1 -norm)

The ℓ_1 -norm, also known as the Manhattan norm, taxicab norm, or Lasso is a mathematical concept used to measure the distance or magnitude of a vector in a space. Unlike the Euclidean norm, which calculates the distance as the square root of the sum of squared components, the ℓ_1 -norm computes the distance as the sum of the absolute values of the vector's individual components.

The term "Manhattan norm" derives from the concept of distance traveled in a city grid, where movements can only occur along perpendicular streets. The ℓ_1 -norm reflects the distance traveled by a taxi navigating city blocks to reach from one point to another, moving horizontally and vertically without diagonal movement. It evaluates the absolute sum of differences along each axis between two points, providing a practical way to measure distance in various scenarios, such as feature selection in machine learning and signal processing.

One notable characteristic of the ℓ_1 -norm is its robustness to outliers compared to the ℓ_2 -norm. When dealing with data containing outliers or extreme values, the ℓ_1 -norm tends to be less affected because it doesn't heavily emphasize large deviations due to its linear nature. This property makes it valuable in scenarios where robustness against outliers is crucial, such as in certain regression models or in optimizing solutions where sparse representations are desired.

The ℓ_1 -norm also plays a significant role in compressed sensing, a signal processing technique that reconstructs signals from sparse observations. In this context, the ℓ_1 -norm acts as a convex optimization criterion that promotes sparsity, aiding in accurately recovering signals from significantly fewer measurements than traditionally required, enabling efficient data compression and reconstruction.

- Zero Norm (ℓ_0 -norm)

The zero-norm serves as a crucial concept in measuring sparsity, quantifying the count of non-zero elements in a vector or signal. Despite its non-convexity and computational challenges, it plays a significant role in understanding and promoting sparsity in various applications, particularly in sparse optimization, feature selection, and compressed sensing. In optimization problems, particularly in ℓ_0 regularization, minimizing the zero-norm of coefficients or parameters encourages sparsity in solutions by penalizing non-zero elements. In machine learning and statistics, the zero-norm assists in feature selection, where it quantifies the number of relevant features contributing to a model. Minimizing the zero-norm for sparsity is challenging due to its non-convex nature, leading to approximations or surrogate penalties like ℓ_1 norm (Lasso) regularization (Ramirez, Kreinovich, & Arguez, 2013).

Sparse solutions often offer more interpretable models, as they highlight the most significant variables or features contributing to the outcome. For instance, in linear regression with Lasso regularization, the non-zero coefficients directly indicate the relevant features influencing the target variable.

Finding sparse solutions can be computationally intensive, especially for large-scale problems or in cases where exact sparsity is hard to achieve. Balancing sparsity and accuracy is crucial. Overemphasizing sparsity might lead to loss of important information, affecting predictive performance. Identifying the appropriate sparsity-inducing regularization and tuning its hyperparameters is essential to achieve the desired level of sparsity without compromising model performance.

Sparse optimization reduces computational burden by simplifying models, making them more scalable and practical for large datasets. Sparse solutions enhance interpretability by highlighting significant variables or features, aiding in decision-making and model understanding. Sparsity helps control model complexity, preventing overfitting and improving generalization to unseen data.

A fundamental gap in the literature on deep learning and the pruning of DNNs is the absence of automated pruning percentage determination. More importantly, the

literature primarily focuses on accuracy, neglecting the inclusion of layers that would provide diversity. In this thesis, which aims to address this gap and need, deep learning layers will be pruned for the first time with an automated optimization algorithm. Simultaneously, the proposed method will satisfy both accuracy and diversity trade-off criteria. The proposed optimization model consists of two stages:

- *Pruning with Sparsity*
- *Accuracy-Diversity Trade-Off.*

Pruning, often referred to as model pruning or sparsity induction, involves reducing the size of a neural network by eliminating redundant or less influential connections, parameters, or neurons. This technique is vital for optimizing models, enhancing computational efficiency, and reducing memory requirements without compromising performance. Neural networks often contain redundant connections or parameters that contribute minimally to the network's overall performance. Pruning aims to identify and remove these redundant elements while preserving or even improving model accuracy. Types of pruning are listed as follows:

- *Weight Pruning*

Weight pruning is a technique used in machine learning and neural network optimization to reduce model size and computational complexity. It involves identifying and eliminating certain connections or parameters (weights) within a neural network that contribute minimally to the network's performance. By removing these less influential connections, weight pruning aims to create a more compact and efficient model without significantly sacrificing its accuracy.

There are various approaches to weight pruning, including magnitude-based pruning, which involves identifying weights with low magnitudes or those close to zero, and setting them to zero or removing them entirely. Another method is structured pruning, which targets entire neurons, channels, or filters based on specific criteria, often eliminating entire sets of weights rather than individual connections. Moreover, iterative pruning techniques progressively prune the network during training, allowing the model to adapt and compensate for removed connections, resulting in a more robust and sparser architecture.

Weight pruning offers several advantages beyond model compression. It reduces memory requirements, making it feasible to deploy models in resource-constrained environments such as mobile devices or edge computing devices. Additionally, pruned models often exhibit faster inference times due to the reduced number of computations

required for forward passes, enabling real-time applications and more efficient utilization of computational resources.

However, while weight pruning can lead to significant model compression, the process of identifying and eliminating weights requires careful consideration. Repeated iterations of pruning and retraining are often necessary to maintain or recover the model's performance. Pruning excessively or without proper fine-tuning can lead to a loss in accuracy or generalization ability. Therefore, a balance between model compression and preserving performance must be struck to derive the full benefits of weight pruning in neural network optimization.

- *Neuron Pruning*

Neuron pruning, a technique in neural network optimization, involves the removal or elimination of entire neurons from a neural network architecture. Unlike weight pruning which targets individual connections or parameters, neuron pruning focuses on removing entire computational units (neurons) along with their incoming and outgoing connections. This technique aims to create a more streamlined network by reducing its complexity and size while maintaining or improving its performance.

Neuron pruning can be approached in various ways, including identifying and removing neurons based on their importance or contribution to the network's output. Methods like activation-based pruning assess the activation patterns of neurons during training and eliminate those that are consistently less active or contribute less to the network's overall output. Similarly, sensitivity analysis techniques evaluate neurons based on their impact on the network's loss function and eliminate less influential neurons.

One of the primary benefits of neuron pruning lies in creating more efficient and interpretable neural network architectures. By removing redundant or less informative neurons, the resulting network becomes more compact, requiring fewer computational resources for inference. Additionally, a pruned network may exhibit improved generalization by reducing overfitting tendencies, as it focuses on essential representations and features within the data.

However, similar to weight pruning, the process of neuron pruning requires careful consideration. Pruning too aggressively or without proper retraining can lead to a loss in the network's expressiveness and performance. Iterative pruning, followed by retraining, is often employed to mitigate potential accuracy drops and enable the network to adapt to the removal of neurons while maintaining its functionality. Neuron

pruning represents a valuable approach in optimizing neural network architectures, contributing to more efficient and effective models in various machine learning applications.

Pruning methods often rely on specific criteria such as weight magnitude, sensitivity analysis, or connection importance metrics (e.g., weight sensitivity, second-order derivatives) to determine which parameters or connections to prune. Hence, pruning reduces the computational burden by reducing the number of computations required during inference and sparse models occupy less memory, making them more practical for deployment on resource-constrained devices. Pruning acts as a form of regularization, preventing overfitting by simplifying the model's structure. There are different techniques in pruning, and these are given as below:

i. *Magnitude-based Pruning*

Magnitude-based pruning is a technique used in neural network optimization that identifies and removes connections or weights with low magnitudes or values close to zero. By setting these weights to zero or eliminating them entirely, magnitude-based pruning reduces the computational complexity and memory footprint of a neural network while aiming to preserve its performance. This method exploits the observation that certain weights in a neural network contribute less to its overall function or have minimal impact on the network's output. Typically applied iteratively, magnitude-based pruning involves ranking the weights based on their magnitude and then removing a certain percentage of the smallest magnitude weights, followed by retraining to fine-tune the network and mitigate any potential loss in accuracy, resulting in a more compact and efficient neural network architecture.

ii. *Iterative Pruning*

Iterative pruning is an optimization technique in neural network refinement that involves a cyclical process of removing connections, neurons, or weights from a network in multiple iterations. This method operates by gradually eliminating less relevant components—such as connections with low weights or neurons with minimal impact—followed by retraining the pruned network to recover its performance. Iterative pruning aims to strike a balance between reducing the network's complexity and preserving its accuracy. By iteratively pruning and fine-tuning, the network adapts to the removed components, allowing for the retention of crucial information while progressively creating a more efficient and compact architecture. This iterative approach often results in networks with improved efficiency, reduced computational

requirements, and maintained or even enhanced performance compared to non-pruned models.

iii. Structured Pruning

Structured pruning is a method used in neural network optimization to remove entire structured units, such as channels, neurons, or convolutional filters, rather than targeting individual weights. Unlike magnitude-based pruning, which focuses on eliminating specific weights, structured pruning aims to remove entire structural components based on certain criteria. By removing entire structures, this technique not only reduces model size but also can lead to more significant computational savings and streamlined architectures. Structured pruning often involves identifying and removing less critical parts of the network based on criteria like activation patterns, sensitivity analysis, or other importance metrics, thus creating more efficient and compact neural network architectures while maintaining or improving performance. This approach aligns well with hardware optimization efforts, enabling more efficient deployment on specialized hardware architectures tailored for neural network inference.

iv. Global vs. Local Pruning

Global Pruning: Global pruning is a technique in neural network optimization where connections or parameters are removed based on their global importance across the entire network. Unlike local pruning, which assesses components within specific layers or regions, global pruning evaluates the overall contribution of parameters or connections to the network's performance. This method involves ranking and removing connections or weights considering their impact on the entire network, aiming to create a more compact architecture while maintaining or improving performance. Global pruning strategies often consider metrics like weight magnitudes, sensitivity analysis, or importance scores obtained from training to determine the connections or parameters that can be pruned, resulting in more efficient and streamlined neural network architectures suitable for deployment in resource-constrained environments.

Local Pruning: Local pruning is a neural network optimization technique that focuses on removing connections, neurons, or parameters within specific layers or regions of the network. Unlike global pruning, which considers the overall network impact, local pruning targets components within localized sections of the neural architecture. By assessing and eliminating less crucial elements within specific layers or regions based

on criteria like weight magnitudes, activations, or importance scores, local pruning aims to enhance the efficiency and computational performance of individual segments while maintaining the network's overall functionality. This approach often leads to more targeted improvements in specific parts of the network, allowing for finer control over model complexity and resource utilization in various neural network architectures.

Excessive pruning can lead to a loss of important information, affecting the model's accuracy or performance. Retraining or fine-tuning pruned networks is essential to recover lost accuracy, requiring careful optimization and training strategies. Designing effective pruning algorithms and strategies while ensuring compatibility with different neural network architectures can be challenging. However, pruned models are more lightweight and computationally efficient, making them suitable for deployment on edge devices, mobile applications, and embedded systems. Pruning optimizes resource utilization, reducing memory requirements and energy consumption in inference tasks while it also facilitates model compression, enabling faster transfer learning and sharing of pre-trained models with reduced sizes.

The accuracy-diversity trade-off in optimization models represents a fundamental balancing act between achieving high accuracy and maintaining diversity or generalization ability within a model. In many cases, optimizing for higher accuracy can inadvertently lead to a reduction in diversity or robustness, causing the model to become overly specialized and perform well only on specific, narrowly defined data points. Conversely, emphasizing diversity might sacrifice some accuracy by focusing on a broader range of scenarios or including more diverse but potentially noisy or less informative data.

Striking the right balance between accuracy and diversity is crucial in building models that generalize well to unseen data while still performing effectively on the target task. Overfitting, a scenario where a model becomes too tailored to the training data and performs poorly on new examples, is often a result of overemphasizing accuracy without considering diversity. Regularization techniques, cross-validation, and ensemble methods are approaches used to navigate this trade-off. Regularization methods penalize overly complex models, encouraging simplicity and thereby potentially enhancing diversity. Cross-validation helps in assessing a model's performance across different subsets of data, providing insights into its generalizability. Ensemble methods combine diverse models to leverage their

collective diversity, often resulting in improved accuracy without compromising diversity.

Ultimately, the optimal point in the accuracy-diversity trade-off heavily depends on the specific problem domain, the nature of the data, and the intended application of the model. Balancing these factors ensures that the model not only performs well on known data but also exhibits robustness and adaptability to unforeseen scenarios, enhancing its practical utility and reliability.

Pruning is a crucial technique in optimizing neural network models by eliminating redundant connections or parameters, reducing computational complexity, and optimizing memory requirements. While it presents challenges related to information loss and recovery of performance, effective pruning strategies are essential for developing efficient and practical models suitable for various real-world applications.

In the first stage, as frequently discussed in recent times in deep learning models and machine learning methods, a sparse optimization model will be proposed with ℓ_0 targeting pruning. However, the zero norm is a non-convex and non-differentiable function. Therefore, various studies have been conducted in the optimization and machine learning literature concerning the convergence of sparse models. One of these is the Multiple Kernel Learning (MKL) method.

Kernel methods are defined as non-parametric and powerful techniques in machine learning and data analysis. Multiple Kernel Learning (MKL), instead of using a fixed kernel, is an algorithm that optimizes linear combinations of multiple candidate kernels and computes sparse optimization to find the optimal weights for each kernel function. MKL has shown good performance in many applications (Suzuki & Tomioka, 2011; Wang, Liu, Japkowicz, & Matwin, 2014) MKL offers three significant advantages. First, it enhances the effectiveness of kernel selection. Second, it improves the performance of kernel methods by learning the optimal combination of multiple heterogeneous kernels or multiple parameters of a single kernel. Third, it can be used to merge data from multiple heterogeneous sources (Mariette & Nathalie, 2018). Additionally, many MKL algorithms come with high computational costs.

In the literature, a new optimization algorithm called SpicyMKL has been proposed for Multiple Kernel Learning (MKL), which is applicable to general convex loss functions and general regularization types. SpicyMKL iteratively solves smooth minimization problems. Consequently, there is no need to internally solve Support

Vector Machines (SVM), Linear Programming (LP), or Quadratic Programming (QP). Experimental results show that the SpicyMKL algorithm is notably faster than existing methods, especially when the number of kernels is large (> 1000) (Suzuki & Tomioka, 2011). Additionally, a framework for parallel multiple kernel learning (PMKL) has been developed using Hybrid Alternating Direction Method of Multipliers (H-ADMM) to integrate MKL algorithms into multi-processor systems. In the literature, the global problem of multiple kernel learning is divided into multiple local problems, each optimized on a single-kernel local processor. H-ADMM is used to coordinate local processors to obtain the global optimal solution (Chen & Fan, 2014). Furthermore, there are studies that combine MKL algorithms with Stacked Generalization (SMKL). The results indicate that SMKL can handle more instances with the same number of kernels or combine more kernels than MKL with the same number of instances (Wang, Liu, Japkowicz, & Matwin, 2014). Multiple kernel learning and weight optimization models are, in fact, equivalent to community learning in terms of selecting candidate learner functions to be included within the community while weighting them.

Another equivalent application to model selection is the problem of feature selection within the data. One commonly used method for this purpose is Principal Component Analysis (PCA). Considering the success and speed of sparse models, it can be expressed by near-zero coefficients when unnecessary and unrelated features are removed from the attributes. Therefore, Sparse Principal Component Analysis (SPCA) has been proposed in the literature. SPCA addresses the problem of finding a sparse coefficient vector that maximizes the variability of data and forms a linear combination of variables within a specific dataset. This model enhances the interpretability of principal components and finds applications in various fields, including genetics and finance (Beck & Vaisbourd, 2018; Beck & Hallak, 2018).

In the literature, a tractable binary convex optimization algorithm has been proposed to solve sparse classification problems. When compared to Lasso with this method, it has been empirically observed that, as the number of examples, denoted as n , increases, the number of correctly selected real features by both methods approaches the actual sparsity, and the computation time of the algorithm decreases when more data is available (Bertsimas, Pauphilet, & Parys, 2021). Additionally, proximal mappings containing group sparsity terms have been investigated, and an efficient method has been found to enable computation. In this study, necessary and sufficient

optimality conditions for the problem have been developed, and a hierarchy has been established between stationary-based and coordinate-wise-based conditions (Beck & Hallak, 2018). Under sparse constraints, three numerical algorithms have been used to find points that satisfy optimality criteria: the iterative hard thresholding method, the greedy method, and partial sparse-simplex methods. The first algorithm is essentially a gradient descent method, while the remaining two algorithms are of the coordinate descent type (Beck & Eldar, 2013).

In recent times, sparse problem minimization and low-degree approximation techniques have found widespread applications in various domains, including computer vision, machine learning, telecommunications, and more (Quach, Duong, Luu, & Bui, 2017). In these types of problems, the presence of a ℓ_0 , which corresponds to achieving sparsity, renders the objective function of the optimization problem non-convex. As a result, the optimization community has put forth various approaches to address non-convex objective functions. One such approach involves the development of Semi-Definite Programming (SDP) relaxation methods, which are grounded in the Difference of Convex (DC) decomposition, primarily aimed at a class of non-convex second-degree constrained problems (Quach, Duong, Luu, & Bui, 2017; Zheng, Sun, & Li, 2011). However, SDP algorithms are known to exhibit sluggish performance and demand substantial memory resources when dealing with high-dimensional datasets. In response, researchers have explored the conversion of SDP algorithms into second-order conic problems to enhance their efficiency in relaxation scenarios.

Second-Order Cone Programming (SOCP) stands as a problem-solving approach that bridges the gap between linear or quadratic programming and Semi-Definite Programming (SDP). Like Linear Programming (LP) and SDP, SOCPs can be effectively addressed through primal-dual interior-point methods. Moreover, various engineering problems can be reformulated as second-order cone problems, as observed by Soma Lobo, Vandenberghe, Boyd, and Lebret in 1998. In current literature, efforts have been directed towards resolving non-convex quadratically constrained quadratic programming (QCQP) problems by introducing novel convex relaxation techniques. The primary semidefinite programming relaxation often proves inadequate for general QCQPs. As a result, recent research has concentrated on enhancing these convex relaxations by making use of valid linear or second-order cone (SOC) inequalities. In these studies, valid second-order cone constraints have been devised for non-convex QCQP, and employing these valid constraints has led to a

reduction in the duality gap, as seen in the work by Jiang, Rujun, and Li in 2016. Additionally, within the literature, a branch-and-bound algorithm has been created for solving continuous optimization problems in which a non-convex objective function is minimized under non-convex inequality constraints. This is feasible when certain solvability assumptions are met, as detailed in the research by Beck and Pan in 2017. The relationship between linear programming (LP), quadratic programming (QP), second-order cone programming (SOCP), semi-definite programming (SDP) and convex programming (CP) are visualized in Figure 5:

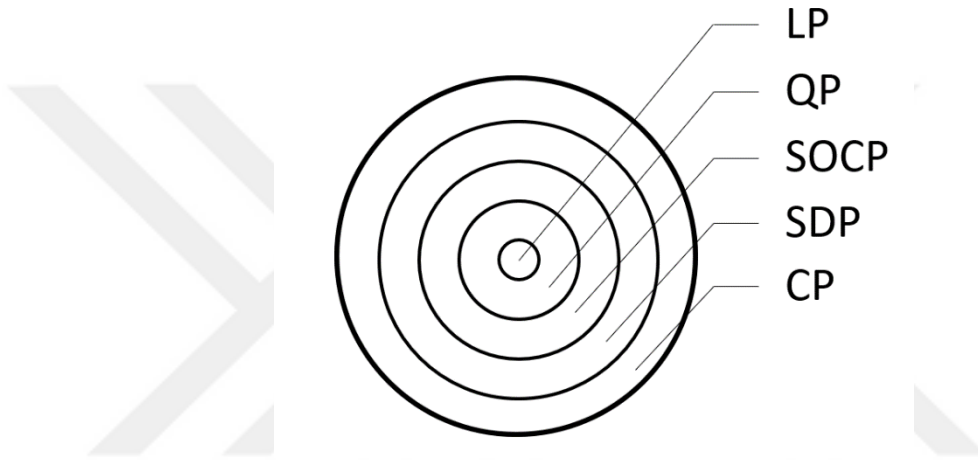


Figure 5 A structured sequence of convex optimization

In another study that aimed to achieve sparsity, a smoothing general optimization approach using the proximal gradient method was employed to handle structured sparsity-inducing penalties by utilizing overlapping group lasso penalties and graph-guided fusion penalties (Chen, Lin, Kim, Carbonell, & Xing, 2012). Similarly, in other related works, proximal mapping was explored in terms of the zero-norm over symmetric sets. The approach, known as Second Order Monotonicity (SOM), was developed for symmetric sets, which exhibit a property akin to sub modularity instead of the traditional second-order monotonicity (Beck & Hallak, 2018).

Semi-definite relaxation methods serve as powerful techniques in optimization, converting challenging non-convex problems into tractable semi-definite programming (SDP) formulations. However, these methods are not without their limitations and drawbacks. One significant drawback of SDP relaxations is their reliance on approximations, leading to potential loss of precision. When converting

non-convex problems into semi-definite programs, approximations or relaxations are often employed to simplify the original problem. While these relaxations facilitate easier computation, they might deviate from the optimal solution, impacting the accuracy of the final result. As a consequence, the solutions obtained through SDP relaxations may not always guarantee the global optimum, especially in highly non-convex scenarios. Another drawback pertains to computational complexity. Solving SDP formulations can be computationally demanding, especially as the size of the problem grows. The optimization of semi-definite programs often requires sophisticated algorithms and substantial computational resources, making them impractical for large-scale problems. The computational burden escalates with the increased dimensions of the semi-definite matrices, resulting in longer processing times and memory requirements. Consequently, while SDP relaxations provide a powerful framework for addressing non-convex problems, their computational demands limit their applicability to smaller-scale or moderately sized optimization tasks. Furthermore, the reliance on semi-definite relaxations introduces conservatism in the solutions obtained. The approximations made during the relaxation process can sometimes lead to overly conservative bounds or estimates. This conservatism may result in overly pessimistic solutions, affecting the model's performance by providing solutions that are overly cautious and far from the true optimal solution. Striking a balance between computational efficiency and solution accuracy becomes a critical challenge in the application of semi-definite relaxation methods, as the trade-off between accuracy and computational tractability must be carefully managed. A drawback of the semidefinite programming relaxation method for second-order or combinatorial optimization problems lies in its expensive computational cost. To address this, an alternative relaxation method, called second-order cone programming (SOCP) relaxation, is proposed by introducing convex valid inequalities for the positive semidefinite cone. These inequalities enhance the lift-and-project linear programming relaxation method, making it more efficient (Kim & Kojima, 2001).

In this thesis, a model will be introduced for pruning an ensemble of different convolutional neural network models using an optimization framework. As evident from the literature summary, sparse models have shown more success and efficiency in optimization studies compared to second-order cone programming SDP. One of the unique contributions of this project is the direct derivation of the pruning percentage parameter, which is vital in the deep learning literature, through the proposed second-

order cone optimization model. Another significant value added by this project addresses the issue of diversity criteria being overlooked when pruning deep learning networks, which are usually pruned based solely on accuracy performance. The objective function in the proposed optimization model will simultaneously optimize both accuracy and diversity criteria to select the best layers. The new deep learning pruning algorithm developed in this research will be tested on publicly available CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and MNIST datasets, and the results will be shared. Additionally, the optimization model defined for pruning also will be adapted to the feature selection problem and compared with various feature selection algorithms in the literature.



Chapter 3

Methods

Ensemble learning stands as a powerful paradigm in machine learning, leveraging the collective wisdom of multiple models to enhance predictive accuracy, generalizability, and robustness. In the realm of deep learning, ensemble methods have emerged as a compelling approach, particularly in the context of Convolutional Neural Networks (CNNs), offering a means to elevate performance and address inherent complexities.

Ensemble methods operate on the principle of combining diverse base learners to create a stronger, more resilient model. In the context of CNNs, ensemble learning manifests in the fusion of multiple CNN architectures or model variations, each capturing distinct patterns or representations within the data. This combination of models can occur through various ensemble strategies such as bagging, boosting, or stacking. Bagging (Bootstrap Aggregating) involves training multiple CNNs on different subsets of the data and combining their predictions, thereby reducing overfitting and enhancing model stability. Boosting, on the other hand, iteratively builds a sequence of CNNs, with each subsequent model focusing on the errors of its predecessor, leading to improved overall performance. Stacking combines the predictions of individual CNNs into a meta-model, leveraging their collective knowledge to generate more accurate predictions. The No Free Lunch Theorem, found in the literature, demonstrates that there is no single model that provides the best solution for every problem (Xu, Caramanis, & Mannor, 2011).

Ensemble methods in deep learning, particularly ensembles of CNNs, offer several advantages. They excel in scenarios where single models might struggle due to complex, high-dimensional data, or limited datasets. By leveraging diverse architectures or training methodologies, ensemble CNNs mitigate the risk of model bias and variance, leading to more robust and accurate predictions. Additionally, ensemble CNNs exhibit enhanced generalizability, performing well on unseen data by leveraging the collective insights of individual models. Moreover, they provide an avenue for addressing interpretability challenges in deep learning models by enabling scrutiny of multiple architectures' collective decision-making processes.

Utilizing ensemble methods within Convolutional Neural Networks (CNNs) presents a series of challenges, despite its potential benefits. One major hurdle is the

considerable computational demand necessitated by training and maintaining multiple models concurrently. This demand for computational resources escalates significantly as more models are introduced into the ensemble. A critical aspect in ensemble construction involves achieving diversity among constituent models while maintaining computational efficiency. Balancing diversity and accuracy within the ensemble becomes paramount to prevent overfitting and to maximize the ensemble's predictive prowess.

The primary goal of ensemble learning is to elevate predictive accuracy beyond the capabilities of a single model. Single classifiers, often constrained by potential noise in the data, overlapping data distributions, and the influence of outliers, can fall short of achieving the desired level of classification accuracy. These limitations underscore the necessity for ensemble techniques, as they offer a means to navigate the complexities and uncertainties inherent in real-world datasets. Ensemble methods effectively mitigate the constraints of individual models, enhancing the overall predictive performance by aggregating diverse perspectives and predictions.

Research studies have highlighted the rising demands for model development to the adoption of ensemble techniques (Dietterich, 2000; Polikar, 2012). These methodologies have become integral due to their ability to address the limitations of individual models, effectively tackling the challenges arising from noise, overlapping distributions, and outliers in datasets. Nonetheless, the implementation of ensemble methods within CNNs necessitates a delicate balance between computational demands and the ensemble's performance, emphasizing the ongoing need for efficient and scalable ensemble strategies. Generation of the ensemble of CNNs involves creating multiple CNN architectures or training the same architecture with different initializations or training data variations, then combining their predictions to make a final decision. There are several approaches to constructing ensembles of CNNs:

- **Bagging (Bootstrap Aggregating):** It involves training multiple CNN models on different subsets of the training data by resampling with replacement. The final prediction is obtained by averaging or voting on the predictions of these individual models.

- **Boosting:** Boosting sequentially builds multiple CNN models, each focusing on the examples the previous models found difficult. Models are weighted based on their performance, and their collective prediction forms the final output.

- **Stacking:** Stacking combines predictions from multiple diverse CNN models

by training a meta-model that learns to make predictions based on the outputs of the individual models.

The advantages of ensemble of convolutional neural networks can be listed as below:

- *Improved Accuracy*: Ensembles often achieve higher accuracy compared to single models, as they reduce overfitting and capture diverse aspects of the data.

- *Enhanced Robustness*: By combining predictions from multiple models, ensembles are less susceptible to noise or outliers present in the data.

- *Better Generalization*: Ensembles generalize well to unseen data, reducing the risk of overfitting to the training dataset.

- *Model Diversity*: Each CNN in the ensemble might specialize in recognizing different features or patterns, collectively improving the overall performance.

The challenges in ensemble of convolutional neural networks are;

- *Computational Resources*: Ensembles typically require more computational resources due to training multiple models.

- *Complexity*: Managing and fine-tuning an ensemble might be more complex compared to a single model.

- *Diversity vs. Accuracy Trade-off*: Ensuring diversity among individual models while maintaining performance can be challenging.

Ensemble learning with CNNs presents a powerful methodology to boost performance, robustness, and generalization in various computer vision tasks. By combining the strengths of multiple CNN models, ensembles continue to push the boundaries of what is achievable in image analysis and recognition tasks, driving advancements across numerous fields.

Traditionally, in the literature, ensembles have been generated using bootstrap aggregation (bagging) techniques. However, recent findings have emphasized that random initialization of deep models within the ensemble is more effective than bagging approaches (Lee, Purushwalkam, Cogswell, Crandall, & Batra, 2015). Ensembles consisting of randomly initialized DNNs explore the function space of architectures more extensively, covering a broader range on the accuracy-diversity spectrum. They also compete effectively with methods such as weight averaging (Izmailov, Podoprikin, Garipov, Vetrov, & Wilson, 2018) and local Gaussian approximations (Nguyen-Tuong, Seeger, & Peters, 2009) which produce models with limited diversity in the function space.

In this thesis, to create an ensemble with the desired diversity and accuracy, we chose to generate 50, 150, and 300 CNN models using a random selection process based on the following design specifications (Güldoğuş, Abdullah, Ali, & Akyüz, 2023):

- *Convolution Blocks*: Every convolution block comprises a pair of convolution layers, a batch normalization layer, a ReLU (Rectified Linear Unit) activation function for non-linear transformations, and a pooling layer. The number of blocks was chosen randomly from a range of 3 to 5.

- *Convolution Filters*: The number of convolution filters in each convolution layer is randomized within a range of 32 to 256, using increments of 32.

- *Pooling Technique*: Two pooling methods are selected randomly from a set that includes average pooling and maximum pooling.

- *Size of the Classification Head (Fully Connected Layer)*: The number of neurons in the fully connected (dense) layer is randomly determined within a range of 20 to 100, with increments of 10.

- *Dropout Layer Rate*: The dropout regularization layer's rate is randomly selected within a range from 0 (no dropout) to 0.5, with increments of 0.1.

- *Learning Rate*: The learning rate of the optimizer was randomly chosen from a range between $1e-4$ and $1e-2$ using a logarithmic scale.

- The ADAM optimizer and the Sparse Categorical Cross-Entropy loss function are utilized for training all CNN models.

3.1 Creating the Optimization Model that Satisfies the Most Suitable Conditions from the Library

Optimization plays a pivotal role in harnessing the true potential of ensemble methods, especially when applied to Convolutional Neural Networks (CNNs). Optimizing an ensemble of CNNs involves various strategies and techniques aimed at improving performance, enhancing diversity among constituent models, and managing computational resources efficiently.

Optimization techniques are crucial for promoting diversity among individual CNN models within an ensemble. Diversity ensures that models capture different aspects or representations of the data, reducing biases and errors that might be present in a single model. Optimization strategies can be listed as follows:

- *Varying Architectures*: Employing different CNN architectures (varying depths, filter sizes, etc.) within the ensemble.

- *Training on Diverse Data:* Introducing variations in training data through techniques like data augmentation, dropout, or using different subsets of the dataset for each model.

- *Hyperparameter Tuning:* Optimizing hyperparameters such as learning rates, regularization strengths, and batch sizes to achieve model diversity.

Pruning the ensemble with an optimization model also involves efficiently managing computational resources without compromising performance. Removing redundant or less-contributing models from the ensemble to reduce computational overload and leveraging parallel computing to train multiple models concurrently, reducing overall training time.

Optimization directly impacts the performance of the ensemble by focusing on strategies that enhance overall predictive accuracy and robustness by employing various combination methods such as averaging, weighted voting, or stacking to merge predictions from individual models effectively and determining the optimal size of the ensemble to balance performance gains with computational costs. Optimization techniques are useful for mitigating overfitting and improving generalization capabilities by applying regularization methods like dropout, weight decay, or early stopping to prevent overfitting of individual models and also utilizing cross-validation techniques to assess the ensemble's performance on different subsets of the data, ensuring robustness and generalization.

Optimization in ensemble learning for CNNs is fundamental for maximizing the benefits of combining multiple models. It not only focuses on improving predictive accuracy but also addresses challenges such as computational efficiency, diversity management, overfitting, and adaptability. By employing an array of optimization strategies, practitioners can harness the full potential of CNN ensembles, achieving superior performance, robustness, and generalization across various computer vision tasks.

In a system consisting of a binary classifier, suppose that classifier i computes the probability of a selected sample belonging to Class 1 as $f_i^{\{1\}}$. To obtain a consensus across all classifiers in this system;

$$f_{ens} = \sum_{i=0}^{M-1} \omega_i f_i^{\{1\}}, \quad (1)$$

a weighted majority voting method has been proposed (Polikar, 2012). In Equation

(1), the ω_i represents weights that indicate the classifier's performance, and these ω_i values are derived from the optimization model introduced later in Equation (2). If we denote the classifiers' output distributions as p_i and $f_{gt} \in \{0,1\}$ as the ground truth, the ensemble's loss function can be formulated as:

$$L_{ens}(f) = \alpha(f_{ens} - f_{gt})^2 + (1 - \alpha) \left(1 - \left(H \left(\sum_{i=0}^{M-1} \omega_i p_i \right) - \sum_{i=0}^{M-1} \omega_i H(p_i) \right) \right). \quad (2)$$

In Equation (2), the initial segment delineates the mistake, and the subsequent part represents the complement of diversity. Minimizing L_{ens} corresponds to enhancing accuracy and diversity, with their balance regulated by α . This Equation (2) can be adapted for multi-class classification situations, where C denotes the number of classes, and $f_{gt} \in \{0,1\}^C$ illustrates the correct classification vector:

$$L_{ens} = \alpha \sum_{j=0}^{C-1} \frac{(f_{ens,j} - f_{gt,j})^2}{C} + (1 - \alpha) \left(1 - \frac{\sum_{j=0}^{C-1} (H(\sum_{i=0}^{M-1} \omega_i p_{i,j}) - \sum_{i=0}^{M-1} \omega_i H(p_{i,j}))}{C} \right). \quad (3)$$

The first part of the Equation (3) represents the accuracy, and the second part corresponds to the $(1 - \text{diversity})$, defined through the Jensen Inequality by following the Shannon Entropy function ($H(x) = -x \log(x)$). The hyperparameter α regulates the balance between accuracy and diversity aspects. Further details on the convex form of Shannon Entropy as given in the next subsection.

3.1.1 Shannon entropy. Entropy is a fundamental concept in physics, information theory, and various scientific fields, representing a measure of disorder or randomness in a system. In thermodynamics, it's associated with the second law, stating that in a closed system, entropy tends to increase over time. This law implies a directionality towards disorder and highlights the arrow of time systems naturally move from ordered states to more disordered ones.

In information theory, entropy denotes the uncertainty or unpredictability of information within a given context. It measures the average amount of information

produced by a stochastic source of data. In a scenario where all outcomes are equally likely, entropy is at its maximum, indicating maximum uncertainty. Conversely, when an outcome is highly predictable, entropy is lower since there's less uncertainty and therefore less information content.

The concept of entropy also extends beyond the realms of physics and information theory, finding applications in diverse fields such as biology, economics, and even psychology. In biological systems, for instance, entropy plays a role in understanding the spontaneous processes that drive living organisms toward equilibrium. It underlines the inevitable tendency of systems to move from order to disorder, shedding light on biological processes' directionality and efficiency.

Overall, entropy serves as a crucial metric for understanding the behavior of systems, whether in explaining the flow of heat and energy in physics, quantifying information in data, or comprehending the dynamics of complex systems in various disciplines, offering insights into the fundamental nature of the universe.

Shannon Entropy is a fundamental concept within information theory that quantifies the uncertainty or randomness in a given set of data or information. Introduced by Claude Shannon in the 1940s, Shannon Entropy is a measure of the average amount of information or surprise associated with the outcomes of a random variable. It provides a principled way to assess the unpredictability or disorder within a system.

In essence, Shannon Entropy captures the information content or uncertainty within a dataset. A higher entropy value implies greater uncertainty or unpredictability, indicating a more diverse or disordered distribution of outcomes. Conversely, lower entropy values correspond to a more predictable or ordered dataset with lesser variability among outcomes.

The usage area of Shannon Entropy is widely spread across various fields, including information theory, machine learning, and signal processing. In machine learning, entropy serves as a crucial metric for decision-making in classification tasks. Within decision trees or ensemble learning methods like Random Forests, entropy is employed as a criterion to evaluate the homogeneity of subsets while partitioning data, aiding in feature selection and model training. Moreover, entropy-based techniques underpin various compression algorithms and data encoding methods, facilitating efficient storage and transmission of information by identifying redundancy and minimizing information loss. Shannon Entropy stands as a foundational concept,

providing a robust framework for quantifying uncertainty and information content, essential in diverse scientific and technological domains.

Shannon entropy is defined for a specific discrete probability distribution; it measures how much information, on average, is required to describe random samples from that distribution. The Shannon Entropy of a function A , considered as entropy, is defined as follows (Shannon, 1948):

$$H(A) = - \sum_{i=1}^n p_i \log p_i. \quad (4)$$

The values denoted by Equation (4) are expressed as a probabilistic measure such that if the equality $p(a_i) = 1/n$ holds for all i values, $H(A) \leq \log n$. Considering convexity, when the second derivative is greater than or equal to zero and $y = \log x$, then:

- $\frac{\partial y}{\partial x} = \frac{\partial(\log x)}{\partial x},$
- $\frac{\partial y}{\partial x} = \frac{1}{x},$
- $\frac{\partial^2 y}{\partial x^2} = \frac{\partial(x^{-1})}{\partial x},$
- $\frac{\partial^2 y}{\partial x^2} = \frac{-1}{x^2}.$

The function $y = \log x$ is convex as the second derivative is less than zero. Therefore, given that the term $-\log x$ in the equation is convex, Jensen's inequality is expressed as follows:

$$-\log \left(\sum_{i=1}^n \frac{1}{p_i} p_i \right) \leq \sum_{i=1}^n \left(-\log \frac{1}{p_i} \right) p_i = -H(A). \quad (5)$$

The validity of the inequality stated in Equation (5) depends on the condition that $p_1 = \dots = p_n$ does not hold and it is certain that H is strictly convex concerning p . Following Jensen's inequality, it can be deduced that if H is convex, then $-H$ is undoubtedly concave. Assuming a function $F[p] = -H(p)$, its functional derivative is $\frac{\delta F[p]}{\delta p(z)}(p(z)) = \log p(z) + 1$ (Baez, Fritz, & Leinster, 2011; Nishiyama, 2020).

3.1.2 Second-order cone programming. Sparse optimization deals with problems where the solution possesses a limited number of non-zero elements. SOCP proves instrumental in such scenarios due to its ability to induce sparsity while providing solutions that satisfy linear and convex constraints. SOCP aids in solving problems with ℓ_1 -norm regularization, promoting sparsity by encouraging many coefficients to

become zero. Besides, SOCP assists in feature selection tasks within machine learning algorithms, aiding in identifying the most relevant features while pruning irrelevant ones. Also, in regression problems, SOCP enables the formulation of sparse regression models that effectively handle high-dimensional data with limited observations.

SOCP techniques extend to structured sparsity where entire groups of variables can be simultaneously set to zero, preserving relationships among variables. Applications in structured output learning problems benefit from SOCP's ability to induce sparsity while maintaining structured relationships between variables. SOCP solvers have demonstrated efficiency in solving large-scale optimization problems compared to other techniques, especially in scenarios where the problems are convex or have specific structural properties. The scalability of SOCP allows it to handle problems with thousands or even millions of variables efficiently, making it suitable for modern high-dimensional data settings. This type of models provides robust solutions, ensuring stability and reliability even in the presence of noisy or uncertain data. The algorithms used in SOCP solvers often exhibit good numerical stability, making them less prone to convergence issues compared to other optimization methods. These methods are versatile and applicable across various domains, including engineering, finance, bioinformatics, and more while also its flexibility allows it to generalize to different types of optimization problems, further expanding its applicability.

Optimization problems aiming to minimize the Cartesian product of second order (Lorentz) cones where they intersect with a linear function and an affine linear manifold are recognized as second-order cone programming (SOCP) problems. Furthermore, linear problems, convex quadratic programs, and second-order cone-constrained convex quadratic programs can all be reformulated as SOCP problems (Alizadeh & Goldfarb, 2003). The problem expressed in Equation (6) is identified as a second-order cone programming problem (Lobo, Vandenberghe, Boyd, & Lebre, 1998):

$$\begin{aligned} \min f^T x \\ \text{st. } \|A_i x - b_i\| \leq c_i^T x + d_i, i = 1, \dots, N, \end{aligned} \tag{6}$$

such that $x \in R^n$ is an optimization variable and the parameters of the problem are; $f \in R^n, A_i \in R^{(n_i-1)n}, b_i \in R^{n_i-1}, c_i \in R^n$ and $d_i \in R$. The norm given in

Equation (6) is the standard Euclidean norm; $\|u\| = (u^T u)^{1/2}$. $\|A_i x = b_i\| \leq c_i^T x + d_i$ is the second-order cone constraint where its dimension is n_i . The k -dimensional unit second-order convex cone is defined as follows:

$$\ell_k = \left\{ \begin{bmatrix} u \\ t \end{bmatrix} \mid u \in R^{k-1}, t \in R, \|u\| \leq t \right\}.$$

Here, when $k = 1$, the unit second-degree cone transforms into the following form:

$$\ell_1 = \{t \mid t \in R, 0 \leq t\}.$$

The set of points that satisfy a constraint of a second-degree cone problem is the inverse image of the unit second-degree cone under an affine mapping:

$$\|A_i x = b_i\| \leq c_i^T x + d_i \leftrightarrow \begin{bmatrix} A_i \\ c_i^T \end{bmatrix} x + \begin{bmatrix} b_i \\ d_i \end{bmatrix} \in \ell_{n_i},$$

Therefore, it is convex (Lobo, Vandenberghe, Boyd, & Lebet, 1998).

3.1.2.1 Duality and complementarity slackness. In second-order cone programming (SOCP), duality and complementarity are essential concepts that contribute to understanding the relationship between primal and dual problems, as well as the optimality conditions. The primal problem aims to minimize a linear objective function subject to second-order cone constraints. The dual problem is derived from the primal and involves maximizing a concave function subject to constraints, where the Lagrange multipliers associated with the primal constraints become variables.

The duality gap represents the difference between the optimal values of the primal and dual problems. Strong duality holds in SOCP when the primal and dual problems have the same optimal value, ensuring that the optimal solutions of both problems exist and are equal. In the dual problem, the Lagrange multipliers associated with the second-order cone constraints in the primal problem become the dual variables. The duality theory provides essential optimality conditions: the primal and dual feasible solutions satisfy the complementary slackness conditions, reflecting a relationship between primal and dual variables.

Duality and complementarity concepts are crucial in developing algorithms for solving SOCP problems, like interior-point methods or dual ascent algorithms. The duality and complementarity aids in sensitivity analysis, evaluating the effects of parameter changes on optimal solutions and their relationships. These concepts provide valuable theoretical insights into the structure of optimization problems,

convergence analysis, and the relationships between primal and dual solutions.

There are similarities to duality theory in linear programming, there exist some distinctions. The standard representation of the second-order cone problem and its corresponding dual problem are detailed below (Alizadeh & Goldfarb, 2003):

Main Problem (P)	Dual Problem (D)
$\min \quad c_1^T x_1 + \cdots + c_r^T x_r$	$\max \quad b^T y$
$\text{st.} \quad A_1 x_1 + \cdots + A_r x_r = b$	$\text{st.} \quad A_i^T y + z_i = c_i, \quad i = 1, \dots, r.$
$x_i \geq Q, \quad i = 1, \dots, r$	$z_i \geq Q, \quad i = 1, \dots, r$

After defining the primal and dual problems, certain assumptions are made:

Assumption 1: The matrix $A = (A_1, \dots, A_r)$ with m rows is linearly independent.

Assumption 2: Both the primal and dual problems contain strongly feasible solutions. For instance, if there exists an $x = (x_1; \dots; x_r)$ vector feasible for the primal problem with $x_i \geq Q, i = 1, \dots, r$, then there also exist dual-feasible vectors y and $z = (z_1; \dots; z_r)$ such that $z_i \geq Q, i = 1, \dots, r$.

Considering the structure as a linear problem:

$$c^T x - b^T y = (y^T A + z^T) x - x^T A^T y = x^T z$$

and according to the self-duality in the cone Q , weak, strong, and semi-strong dualities are defined as follows:

Weak Duality: If x is any feasible solution of the primal problem and (y, z) is a feasible solution of the dual problem, then the duality gap $c^T x - b^T y = z^T x > 0$ holds.

Strong Duality Theorem: If both the primal and dual problems have strictly feasible solutions $(x_i > Q, z_i > Q, \forall_i)$, then the optimal solutions $x^*, (y^*, z^*)$ and $p^* = c^T x^* = d^* = b^T y^*, (x^{*T} z^* = 0)$ are attained for both problems.

Semi-Strong Duality Theorem: If the primal problem (P) has a strictly feasible point $\hat{x}$ in the interior of the feasible region $\hat{x} \in \text{int}(Q)$, and its objective function is bounded from below within the feasible region, then the dual problem (D) is solvable, and $p^* = d^*$. This theorem and the semi-strong duality are applicable to all closed, pointed, convex, and full-dimensional cones in the realm of conic linear problems and have been proven using *Farkas' Lemma* (Nesterov & Nemirovski, 1994).

In linear problems, the complementary slackness conditions imply that if $x \geq$

0 and $z \geq 0$, the duality gap $x^T z = 0$ holds when the condition $x_i z_i = 0$ is satisfied. In second-order cone problems, due to the self-duality of the second-order cone itself, geometrically $x^T z = 0$.

Complementary Conditions:

These conditions relate to the optimal solutions of the primal and dual problems:

- If a primal constraint is active (binding), its associated dual variable must be non-zero.

- If a dual variable is non-zero, the corresponding primal constraint must be active.

The complementary slackness conditions illustrate the "complementarity" between primal and dual variables. Complementarity reveals a relationship where the optimality of one problem's solution implies the optimality of the other. In SOCP, it helps identify active constraints and understand the structure of the solution space. Complementarity ensures that a feasible solution satisfies both the primal and dual problems and that an optimal solution adheres to both the optimal primal and dual solutions.

Assuming hypothetically; if $x \in Q$ and $z \in Q$ then $x^T z = 0$.

$$x_i^T z_i = x_{i0} z_{i0} + \bar{x}_i^T \bar{z}_i = 0, \quad i = 1, \dots, r$$

$$x_{i0} \bar{z}_i + z_{i0} \bar{x}_i = 0, \quad i = 1, \dots, r.$$

Combination of the feasibility of primal and dual problems with the above complementary conditions, the optimality conditions are defined (Alizadeh & Goldfarb, 2003).

Optimality Conditions: If strong feasible solutions exist for the primal problem and the dual problem, the following conditions hold for an optimal solution pair $(x, (y, z))$:

$$Ax = b, \quad x \in Q,$$

$$A^T y + z = c, \quad z \in Q,$$

$$x \circ z = 0.$$

If $r = 1$, it means there is only one block, and the above equations can be solved analytically. If $b = 0$, it is assumed that $\bar{A}$ is of full rank, $x = 0$, and the dual pair (y, z) is the optimal primal-dual solution pair. Additionally, if $c \in \text{Span}(A^T)$, then $A^T y = c$ for y , subsequently $z = 0$, and for any feasible x of the primal problem, it becomes the optimal primal-dual pair. Considering $A = (a, \bar{A})$ is of full rank, there are two cases

to consider:

○Case 1: If ARA^T is nonsingular; by replacing z with αRx in the second equation of the system given above, and solving the resulting system, x and y ;

$$y = (ARA^T)^{-1}(ARc - \alpha b),$$

$$x = \frac{1}{\alpha} P_R c + RA^T (ARA^T)^{-1} b$$

where $P_R = R - RA^T (ARA^T)^{-1} AR$. Then, by using $0 = z^T x = \alpha x^T R x$ the following equation is obtained;

$$\alpha = \sqrt{\frac{-c^T P_R c}{b^T (ARA^T)^{-1} b}}.$$

If $A^+ = A^T (AA^T)^{-1}$ is the pseudo-inverse of A and P is the orthogonal projection operator in R^n reflecting all vectors onto the kernel of A , given as $P = I - A^+ A$ in an orthogonal manner:

$$(ARA^T)^{-1} = -(AA^T)^{-1} - \frac{2}{\gamma} (AA^T)^{-1} a a^T (AA^T)^{-1}$$

$$P_R = RP - \frac{2}{\gamma} ((A^+ a)(A^+ a)^T - A^+ a e^T),$$

only when ARA^T is nonsingular and $\gamma \neq 0$, where $\gamma = 1 - 2a^T (AA^T)^{-1} a$. Manipulating the algebraic expressions provided for $(ARA^T)^{-1}$ and P_R above:

$$y = (AA^T)^{-1}(Ac + \alpha b - \delta a),$$

$$\alpha Rx = z = c - A^T y = P_c - A^+(\alpha b - \delta a),$$

where α and δ are defined as follows:

$$\alpha = \sqrt{\frac{-\gamma c^T P_c + 2(e^T P_c)^2}{\gamma b^T (AA^T)^{-1} b + 2(a^T (AA^T)^{-1} b)^2}},$$

$$\delta = \frac{c^T P_c + \alpha^2 b^T (AA^T)^{-1} b}{e^T P_c + \alpha a^T (AA^T)^{-1} b}.$$

○Case 2: If ARA^T is singular; $\gamma = 1 - 2a^T (AA^T)^{-1} a = 0$ and ARA^T has a one-dimensional kernel N . The followings are defined; $v \in N$, $v \neq 0$, $u = RA^T v \neq 0$, $A^T u = 0$, $u^T R u = v^T A u = 0$. Therefore, if the sign of v is chosen correctly and the corresponding u is both in the kernel of A and on the boundary of Q . Here, $z = \alpha Rx$, y and z satisfy the equations in non-singular conditions, but as $c_0 = a^T y + z_0 = a^T (AA^T)^{-1}(Ac + \alpha b) + z_0 \gamma$ and $\gamma = 0$;

$$\alpha = \frac{e^T P_c}{a^T (AA^T)^{-1} b},$$

$$\delta = \frac{c^T P_c + a^2 b^T (AA^T)^{-1} b}{2e^T P_c}.$$

The provided definitions of δ in both cases are actually the same. Therefore, these same equations can also be used when ARA^T is singular (Alizadeh & Goldfarb, 2003).

3.2 Redefining the proposed loss function as Second-Order Cone

Programming

Convex quadratic functions represent a critical class of mathematical functions with distinctive properties that make them particularly significant in optimization, economics, engineering, and various scientific fields. A convex quadratic function is a quadratic equation that satisfies the criteria of convexity, exhibiting a U-shaped curve or bowl-shaped surface when plotted in multi-dimensional spaces.

The convexity of quadratic functions is characterized by their second-order behavior, where the Hessian matrix, represented by Q in the quadratic function, plays a pivotal role. A key feature of convex quadratic functions is that their Hessian matrix is positive semi-definite, ensuring that the function has a single global minimum or maximum, depending on whether it is a convex or concave function. This property makes convex quadratic functions particularly amenable to optimization, as algorithms can efficiently converge to the global optimum without getting stuck in local minima.

Moreover, convex quadratic functions possess well-defined geometric properties that facilitate efficient optimization techniques. The presence of a single global minimum or maximum enables convex quadratic functions to be solved analytically or through various optimization algorithms with guaranteed convergence to the optimal solution. Techniques like gradient descent, Newton's method, or conjugate gradient methods are effective for minimizing or maximizing convex quadratic functions due to their smooth and well-behaved nature, ensuring rapid convergence and facilitating efficient computation. The pervasive presence of convex quadratic functions across optimization problems underscores their significance as fundamental building blocks for modeling various real-world phenomena and as cornerstone elements in optimization algorithms.

Since, any local minimum within a convex framework also represents a global minimum, and an approximation of the global minimum can be calculated. These problems can be converted into second-order cone problems and resolved through the application of the primal-dual interior point method (Kim & Kojima, 2001). Typically,

convex quadratic models are articulated as outlined in Equation (7).

$$\begin{aligned}
& \min_x \quad x^T Q x + c^T x \\
& \text{st.} \quad Cx \leq d \\
& \quad \quad Ax = b
\end{aligned} \tag{7}$$

where $Q = Q^T \geq 0$. If a quadratic term is added to the objective function, it transforms into the following form:

$$\begin{aligned}
& \min_{x,t} \quad t + c^T x \\
& \text{st.} \quad Cx \leq d \\
& \quad \quad Ax = b \\
& \quad \quad t \geq x^T Q x.
\end{aligned} \tag{8}$$

It can be observed that the additional constraint can be expressed by using a rotated second-order cone. Mathematically, the inequality $t \geq x^T Q x$ only holds when $(Q^{\frac{1}{2}}x, t, 1) \in Q_{rot}^n$.

The loss function defined in Equation (3) can be written as below in a more closed form:

$$\min_x f(x) + \lambda \|x\|_0. \tag{9}$$

The function $f(x)$ in Equation (9) has become a convex quadratic function by employing Shannon Entropy function in a convex form. When applied to deep networks, datasets are expected to have high correlations and dimensions, leading to significant errors with even small changes in the dataset, a phenomenon known as overfitting. To avoid this problem, a regularization term is applied to the problem, aiming to reduce memory usage and model size. The ℓ_0 is used to enforce sparsity, penalizing the number of non-zero entries in the coefficient vector. Ideally, the regularization term is expected to reflect the non-zero entries of the coefficient vector, which typically corresponds to the vector's ℓ_0 -norm. However, the ℓ_0 -norm is not a valid norm because it is non-homogeneous. When non-convex expressions are present in the main problem, finding solutions becomes considerably challenging and these types of problems defined as NP-Hard. The relaxation of the zero norm refers to the process of approximating or transforming the non-convex and non-differentiable zero-norm function into a convex and differentiable form that enables tractable optimization. The zero norm, often denoted as ℓ_0 , counts the number of non-zero

elements in a vector x , rendering it highly non-convex due to its discontinuity and non-smoothness. While the zero norm is a powerful tool for promoting sparsity in machine learning and signal processing by encouraging solutions with fewer non-zero elements, its non-convex nature presents challenges in optimization tasks. To address the computational complexity arising from the non-convexity of the zero norm, relaxation techniques aim to approximate it using convex surrogates that maintain some characteristics of sparsity-inducing norms while enabling efficient optimization. Commonly employed relaxations include the use of the ℓ_1 -norm, also known as the sum of absolute values, which closely approximates the zero norm while ensuring convexity and differentiability. By replacing the zero norm with its convex surrogate, problems involving sparsity constraints become amenable to a wide range of convex optimization algorithms, facilitating the derivation of computationally efficient and globally optimal solutions.

Relaxation of norms in optimization refers to a technique where constraints or conditions within an optimization problem are loosened or relaxed to enable easier or more efficient computation of a solution. Optimization problems often involve strict constraints that make finding an exact solution computationally challenging. By relaxing these constraints, the problem becomes more tractable without significantly compromising the quality of the solution.

In various optimization algorithms, relaxation of norms can take different forms. For instance, in linear programming, constraints may be relaxed to allow a wider feasible region for solutions. Instead of strictly adhering to all constraints, the problem might permit slight violations of certain conditions to explore a broader solution space. This relaxation can lead to faster convergence or even reveal previously unseen feasible solutions.

Moreover, in mathematical optimization, norms are often used to measure the size or distance of a vector or solution from a reference point. Relaxing these norms involves tolerating larger deviations or distances from the optimum. This flexibility allows optimization algorithms to explore a larger search space, potentially discovering better solutions that might have been overlooked under stricter conditions.

The relaxation of norms isn't without trade-offs. While it may expedite the optimization process and uncover alternative solutions, it could also lead to solutions that are farther from the ideal or violate the original problem's constraints. Hence, the balance between relaxation and preserving the problem's integrity is crucial in ensuring

that the optimized solution remains practically viable and acceptable. Overall, the strategic relaxation of norms plays a significant role in enhancing the efficiency and effectiveness of optimization algorithms, especially in scenarios where strict adherence to constraints poses computational challenges.

The reformulation of the zero norm plays a pivotal role in various fields, especially in compressed sensing, sparse signal recovery, and feature selection, where encouraging sparsity is essential. Convex relaxations allow for the formulation of tractable optimization problems that involve sparsity-inducing norms, fostering the development of efficient algorithms capable of handling high-dimensional data or sparse representations. While the relaxed ℓ_1 -norm does not precisely replicate the zero norm's sparsity-promoting capabilities, it serves as a practical compromise, enabling the application of powerful optimization techniques to problems where exact zero norm optimization might be infeasible due to computational complexity. Consequently, the relaxation of the zero norm bridges the gap between computational efficiency and the sparsity-inducing properties required in various real-world applications. Therefore, the literature indicates that the best convex approximation for the ℓ_0 -norm is the ℓ_1 -norm (Ramirez, Kreinovich, & Argaez, 2013). The parameter λ in the equation is introduced as the regularization term. Hence, although the function $f(x)$ is differentiable and convex, the ℓ_0 -norm is not convex. Instead, it is relaxed to the ℓ_1 -norm, offering the best approach when mitigating the ℓ_0 -norm:

$$\min_x f(x) + \lambda \|x\|_1, \quad (10)$$

which transforms into minimization of a convex quadratic problem. Second-order cone rarely appear in the model naturally; thus, a reformulation is a necessity. The norm is relaxed as follows:

$$\|x\|_1 = \sum_{i=1}^n |x_i| \leq u,$$

$$(x_i) \in K_q^2, i = 1, \dots, n.$$

Based on this information, the norm expression is updated in Equation (10), it transforms to the Equation (11);

$$\min_x f(x) + \lambda u \quad (11)$$

$$\text{st. } \sum_{i=1}^n |x_i| \leq u$$

$$x \in K.$$

Here, K is a convex cone such that $K = K^{n_1}x \dots xK^{n_r}xR^{n_i}$ and $x_i \in K^n$ where K^{n_i} is defined as a second-order cone with a dimension n_i as follows;

$$K^{n_i} = x_i = \begin{bmatrix} x_{i_1} \\ x_{i_0} \end{bmatrix} \in R^{n_i-1}xR: \|x_{i_1}\|_2 \leq x_{i_0},$$

where $\|x_{i_1}\|_2$ is the standard Euclidean norm. The convex quadratic second-order cone programming problem with bounded constraints can be seen as a trust-region method, which is a nonlinear programming problem used as a trust-region subproblem (Kato & Fukushima, 2007; Zhang, Liu, & Liu, 2012). Especially in the case of strongly convex quadratic problems, they exhibit polyhedral feasible regions like linear problems and can be expressed in the form of second-order cone problems;

$$\min q(x) = x^T Qx + a^T x + \beta, \quad (12)$$

$$Ax = b$$

$$x \geq 0,$$

the term Q in Equation (12) is a symmetric positive definite matrix; $Q > 0, Q = Q^T$. The main problem $q(x) = \|\bar{u}\|^2 + \beta - \frac{1}{4}a^T Q^{-1}a$ and $\bar{u} = Q^{\frac{1}{2}}x + \frac{1}{2}Q^{-\frac{1}{2}}a$. Therefore, the problem can be written in the SOCP form as Equation (13):

$$\min u_0 \quad (13)$$

$$\text{st. } Q^{1/2}x - \bar{u} = \frac{1}{2}Q^{-1/2}a$$

$$Ax = b$$

$$x \geq 0, (u_0; \bar{u}) \geq Q_0.$$

Both problems yield the same optimal solution; however, there is a change in the main problem by an amount of $\beta - \frac{1}{4}a^T Q^{-1}a$ (Alizadeh & Goldfarb, 2003). Convex quadratic problems are expressed as minimizing a linear function subject to convex quadratic constraints;

$$\min c^T x \quad (14)$$

$$\text{st. } q_i(x) = x^T B_i^T B_i x + a_i^T x + \beta_i \leq 0, \quad i = 1, \dots, m.$$

Here, $B_i \in R^{k_i \times n}$ and rank is $k_i, i = 1, \dots, m$. The constraint $q(x) = x^T B^T Bx + a^T x + \beta \leq 0$ and $(u_0; \bar{u}) \geq Q_0$ which is given as a second-order cone bound express

the same information;

$$\bar{u} = \left(\frac{Bx}{\frac{a^T x + \beta + 1}{2}} \right)$$

$$u_0 = \frac{1 - a^T x - \beta}{2}.$$

Based on the given information, we can rewrite $f(x)$ function as follows by excluding the norm term to add in next steps;

$$\min f(x) = (\alpha)(x^T Qx) + (1 - \alpha)(c^T x), \quad (15)$$

if we define an additional term t where $t = x^T Qx$;

$$\min \quad t + c^T x \quad (16)$$

$$\text{st.} \quad x^T Qx \leq t,$$

and if the constraints are redefined accordingly;

- $t \geq x^T Qx$
- $0 \geq x^T Qx - t$
- $0 \geq 4x^T Qx - 4t$
- $0 \geq 4x^T Qx + (1 - t)^2 - (1 + t)^2$
- $(1 + t)^2 \geq 4x^T Qx + (1 - t)^2$ since $t \geq 0$ and additional

constraint $1 + t \geq 0$ can be added and when Cholesky factorization $Q = LL^T$ is used (Higham, 1990):

- $1 + t \geq \sqrt{4x^T Qx + (1 - t)^2}, 1 + t \geq 0$
- $1 + t \geq \sqrt{4x^T LL^T x + (1 - t)^2}, 1 + t \geq 0$
- $1 + t \geq \left\| \begin{bmatrix} 2Lx \\ 1 - t \end{bmatrix} \right\|, 1 + t \geq 0$
- $\begin{bmatrix} 1 + t \\ 2Lx \\ 1 - t \end{bmatrix} \geq 0.$

Therefore, the problem can be written as a second-order cone problem by also adding the relaxed form of the norm in Equation (17):

$$\begin{aligned} \min \quad & t + c^T x + \lambda u \\ \text{st.} \quad & \begin{bmatrix} 1 + t \\ 2Lx \\ 1 - t \end{bmatrix} \geq 0 \\ & \sum_{i=1}^n |x_i| \leq u \\ & x \in K. \end{aligned} \quad (17)$$

If the Equation (17) is updated with the α coefficients that allow trade-offs in the original problem, the problem is written as follows:

$$\begin{aligned}
& \min \quad (\alpha)(t) + (1 - \alpha)(c^T x) + \lambda u \\
& \text{st.} \quad \begin{bmatrix} 1 + t \\ 2Lx \\ 1 - t \end{bmatrix} \geq 0 \\
& \quad \sum_{i=1}^n |x_i| \leq u \\
& \quad x \in K,
\end{aligned} \tag{18}$$

where $K = K^{n_1} \dots K^{n_r} R^{n_i}$. Here, K^{n_i} is a second-order cone or $K_r^{n_i}$ is a rotated second-order cone and $(x_i) \in K_q^2$. In the Equation (18), t is an additional term denotes to the convex quadratic part while $c^T x$ signifies the linear part where λ is utilized as a regularization term and u is a bound from the reformulation of ℓ_1 norm (Güldoğuş, Abdullah, Ali, & Akyüz, 2023). This proposed model will be adapted to prune the ensemble of CNNs with different depths and sizes on various datasets given below and the pseudocode of this method is given in **Hata! Başvuru kaynağı bulunamadı.** and the results will be given in Section 4.1.

The selected datasets are well-known datasets in the field of machine learning, namely; *CIFAR-10* (Krizhevsky, 2009), *CIFAR-100* (Krizhevsky, 2009), and *MNIST* (LeCun, Denker, & Solla, Optimal Brain Damage, 1990) datasets. Information regarding these datasets is provided below:

- The CIFAR-10 dataset consists of 60,000 32x32 color images divided into 10 classes, with 6,000 images per class. It comprises 50,000 training images and 10,000 test images.
- The CIFAR-100 dataset, similar to CIFAR-10, except it contains 100 classes, each with 600 images. There are 500 training images and 100 test images per class. The 100 classes in CIFAR-100 are divided into 20 super classes.
- The MNIST dataset consists of hand-written grayscale digits commonly used for classification. The MNIST database contains 60,000 training images and 10,000 test images.

CIFAR-10, CIFAR-100, and MNIST are benchmark datasets widely preferred in the realm of machine learning for their distinct characteristics, serving as touchstones for evaluating and benchmarking various algorithms and models. These datasets hold immense significance due to their standardized nature, well-defined

labels, and representation of real-world scenarios in a controlled environment.

MNIST, one of the most renowned datasets, comprises a collection of 28x28 grayscale images of handwritten digits from 0 to 9. It serves as a foundational dataset for beginners due to its simplicity and ease of understanding, making it an excellent entry point for learning classification tasks. The highest test accuracy on MNIST is 99.87% (Byerly, Kalganova, & Dear, 2021). MNIST's small image size and limited complexity facilitate rapid prototyping and algorithm development, allowing practitioners to experiment with different architectures and techniques efficiently.

In contrast, CIFAR-10 and CIFAR-100 present more complex challenges. CIFAR-10 contains 60,000 32x32 color images distributed across ten classes, such as airplanes, automobiles, birds, cats, and others while the highest recorded test accuracy is 99.90% by Google Research Brain Team (Dosovitskiy, et al., 2021) . CIFAR-100 extends this complexity further, encompassing 100 classes with finer-grained distinctions, including subclasses like different breeds of dogs or birds and the highest known test accuracy is generated by Google Research Team (Foret , Kleiner, Mobahi, & Neyshabur, 2021). These datasets emulate real-world complexities more closely, reflecting diverse object classes with higher image resolution compared to MNIST. Their complexity poses a greater challenge for machine learning models, necessitating deeper architectures and more sophisticated algorithms for effective classification.

The preference for these datasets in machine learning lies in their utility as standard benchmarks for evaluating and comparing algorithms. They facilitate a fair and consistent evaluation of models across different research works, enabling researchers to gauge algorithmic performance under various conditions. Moreover, these datasets allow the development and comparison of models' generalizability, robustness, and efficiency in handling real-world image classification tasks.

Another significant factor contributing to their popularity is their accessibility and widespread usage within the research community. Availability in public repositories and integration into popular machine learning libraries like *TensorFlow* and *PyTorch* fosters ease of access and encourages researchers, practitioners, and learners to use these datasets as a common ground for experimentation and innovation.

Furthermore, these datasets aid in the transfer of research outcomes to real-world applications. By testing and validating models on these datasets, researchers can gain insights into how effectively their algorithms can perform tasks like image classification in practical scenarios, paving the way for applications in fields like

computer vision, autonomous vehicles, and healthcare diagnostics.

In conclusion, the preference for CIFAR-10, CIFAR-100, and MNIST in machine learning stems from their standardized, representative nature, enabling fair comparisons and evaluations of algorithms. Their diverse complexity levels cater to various skill levels and research needs, making them indispensable tools for advancing the field of computer vision and machine learning.

The ensemble CNN models were generated using established datasets following standard procedures for training, validation, and testing. The predicted classes and their corresponding probabilities were collected to form the data for the subsequent SOCP process. For both CIFAR-10 and CIFAR-100, 40,000 samples were allocated for training and 10,000 for validation. Meanwhile, the MNIST dataset was split into 40,000 samples for training and 20,000 for validation. All test sets consisted of 10,000 samples. Below is the pseudo code detailing the proposed ensemble pruning method through SOCP modeling in Algorithm 1.

Algorithm 1 *Pruning the ensemble of CNNs with SOCP*

Input: D : dataset, M : number of Convolutional Neural Network Models

Parameters: b : Convolutional Blocks, v : Convolutional Filters, c : Pooling Mechanism, s : Fully Connected Layer Size, d : Drop-Out Layer Percentage, r : Learning Rate, h : sensitivity threshold, l : λ values, a : α values and e : experiment size which is $l.a$

Output: P : Accuracy of Pruned Ensemble

for $i \leftarrow 1$ **to** M **do**

 Split D for training $D_{training}$, validation D_{valid} testing $D_{testing}$

 Randomize b, v, c, s, d and r

 Train with Adam Optimizer and Sparse Categorical Loss Function Save M & $M = 50, 150$ and 300 in this thesis.

 Predict($D_{training}$)

 Save Predictions[l, p] & l = labels, p = probability distributions

end

Use [l, p], $D_{training}$ and $D_{testing}$ as inputs to SOCP model & Please refer to the Equation (18)

Solve SOCP and get $w_{M,e}$ where w =weight vectors

Prune w with h on d_{valid}

Apply Voting Get P .

3.3 A New Feature Selection Technique: Feature Selection via Second-Order Cone Programming – FSOCP

The landscape of feature selection methods spans a rich and diverse array of techniques, each aimed at distilling the most informative features from complex datasets. A brief exploration of the literature reveals a multitude of approaches, including filter, wrapper, and embedded methods. Filter methods operate independently of the learning algorithm, evaluating features based on statistical measures like correlation, mutual information, or statistical significance. These techniques prioritize features solely on their intrinsic properties, often pre-processing the data before model training. While efficient and computationally inexpensive, filter methods might overlook feature interactions and dependencies within the data, potentially selecting irrelevant features.

Conversely, wrapper methods assess feature subsets' efficacy by incorporating specific learning algorithms into the selection process. These methods utilize the learning algorithm's performance on different feature subsets to determine the optimal set for the model. While wrapper methods often yield superior model performance by considering feature interactions, they are computationally intensive and prone to overfitting, especially with large feature spaces.

Embedded methods seamlessly integrate feature selection into the model-building process itself. These techniques select features during model training, optimizing both feature selection and model fitting simultaneously. By considering feature importance within the learning process, embedded methods tend to be more robust and performant. However, they might exhibit limitations in terms of flexibility, as the feature selection process becomes intricately entwined with the model architecture, potentially limiting the model's adaptability to different datasets or tasks. Overall, the choice among these feature selection methodologies hinges on considerations of computational efficiency, model performance, and the specific characteristics of the dataset at hand. Amidst this diverse environment of methodologies, the Equation (18) introduces a novel paradigm, employing a second-order cone programming model to tackle the feature selection enigma. This innovative approach marks a departure from conventional strategies, harnessing the power of optimization within the feature selection, promising a more systematic and efficient way to extract essential features from complex datasets.

The application of the second-order cone programming model, as encapsulated

in the Equation (18), introduces a new path in the feature selection. Unlike conventional methods that often face challenges in scalability and computational complexity, this model presents a structured, mathematically grounded framework. Its integration within the feature selection problem signifies a departure from heuristic-based approaches, promising a systematic and rigorous methodology for identifying crucial features within datasets. The utilization of second-order cone programming in the Equation (18) holds the potential to redefine the boundaries of feature selection by offering an optimization-based solution that ensures robustness, efficiency, and interpretability in the process of distilling meaningful features for model development and analysis.

3.3.1 Background materials. Feature selection is a fundamental process in machine learning and data mining, serving as a critical precursor to model training. Its primary goal is to identify and choose a subset of relevant features from a larger pool of attributes or variables within a dataset. This process is important to enhance model performance by reducing computational complexity and improving the interpretability of learned models. In essence, feature selection, prunes features to extract the most informative and discriminative ones, thereby refining the dataset for model building.

The importance of feature selection stems from its ability to address the curse of dimensionality. In high-dimensional spaces, where the number of features far exceeds the number of observations, learning algorithms face challenges in extracting meaningful patterns due to increased sparsity and computational inefficiency. Feature selection tackles this issue by reducing the number of dimensions, thus mitigating the adverse effects of the curse of dimensionality.

There exist various techniques for feature selection, broadly categorized into three main types: filter methods, wrapper methods, and embedded methods. Filter methods assess the relevance of features based on statistical measures, such as correlation, mutual information, or significance tests, independent of the learning algorithm. Wrapper methods, on the other hand, employ a specific machine learning algorithm to evaluate feature subsets by assessing their impact on model performance. Embedded methods incorporate feature selection within the model building process itself, allowing the algorithm to automatically determine the most relevant features during training.

The choice of feature selection technique depends on several factors, including the nature of the dataset, the computational resources available, and the specific

learning algorithm being employed. Each technique possesses its strengths and weaknesses; therefore, a thorough understanding of their characteristics is crucial in selecting the most appropriate method for a given problem.

One significant challenge in feature selection is striking the right balance between dimensionality reduction and preserving critical information. Selecting too few features may result in loss of important details and underfitting, while an excessive number of features can lead to overfitting, where the model performs well on training data but poorly on new, unseen data. This trade-off requires careful consideration to achieve an optimal balance.

The application of feature selection extends across diverse domains, including but not limited to healthcare, finance, natural language processing, and image recognition. In healthcare, for instance, feature selection aids in identifying crucial biomarkers or health indicators, enabling better diagnosis and treatment planning. In financial analysis, it assists in identifying key factors influencing market trends or investment decisions.

Looking ahead, the evolution of feature selection techniques continues to be a vibrant area of research. With the advent of big data and complex datasets, novel approaches integrating advanced algorithms, such as deep learning and reinforcement learning, are being explored to handle high-dimensional data more effectively. Moreover, efforts are ongoing to develop more robust and scalable feature selection methods that can adapt to evolving data landscapes and diverse machine learning applications.

Feature selection stands as a cornerstone in machine learning, offering a means to improve model performance, reduce computational overhead, and enhance interpretability. As datasets grow in complexity and size, the refinement and selection of relevant features will remain crucial for unlocking the potential of machine learning in various real-world applications. The feature selection methods are discussed under these headings in this thesis (Li, et al., 2018);

- Similarity Based Methods
- Sparse Learning Based Methods
- Statistical Based Methods
- Wrapper Methods

3.3.1.1 Similarity based methods. In feature selection algorithms, various metrics such as distance, separability, information, correlation, dependency, and reconstruction error are utilized to express feature relevance (Cheng, et al., 2017). Methods exploring the significance of features by maintaining the internal similarity of data as constant as possible are termed similarity-based feature selection methods. While supervised feature selection relies on label information for data similarity, unsupervised feature selection techniques leverage different distance measurements for data similarity (Cheng, et al., 2017).

Similarity-based feature selection methods operate on the premise of evaluating feature relevance by measuring similarities or relationships among features within a dataset. These methods leverage metrics or similarity measures, such as correlation, mutual information, or distance-based calculations, to quantify the associations between features. By assessing the interrelationships among attributes, similarity-based techniques identify redundant or highly correlated features, aiming to retain the most informative and diverse subset for model training. These similarity-based methods offer an intuitive approach to feature selection by leveraging the inherent relationships within the dataset. They provide valuable insights into feature interdependencies, aiding in the identification of informative attributes while mitigating redundancy, thereby enhancing model interpretability and performance. However, the effectiveness of similarity-based feature selection methods relies heavily on the choice of similarity measures and their suitability for specific datasets and learning tasks. Careful consideration of the dataset's characteristics and the nature of relationships between features is essential to optimize the efficacy of these methods in extracting the most relevant subset of attributes for downstream analysis.

Consider a dataset $X \in \mathbb{R}^{n \times d}$ with N samples and d features. Pairwise similarity among samples is defined in an affinity matrix $S \in \mathbb{R}^{n \times n}$. The symmetric affinity matrix's entry (i, j) represents the similarity between i^{th} sample x_i and j^{th} sample x_j , where an increase in the value of $S(i, j)$ indicates higher similarity between x_i and x_j . For selecting k strong features from F that yield optimal performance:

$$\max_F \sum_{f \in F} SC(f) = \max_F \sum_{f \in F} \hat{f}' \hat{S} \hat{f}, \quad (19)$$

where $SC(f)$ denotes an auxiliary function for the feature f , and $\hat{S}$ represents the result of the transformation of the original feature vector f (e.g., scaling, normalization, etc.).

$\hat{S}$ is an updated affinity matrix derived from the original affinity matrix S . The maximization problem in Equation (19) indicates the need to select a subset of features from F to preserve the data similarity defined in S as much as possible. This problem can be addressed by selecting the top-k features that maximize their individual benefits $\hat{f}'^{\hat{S}}\hat{f}$. The methods in this category can vary according to the similarity matrix $\hat{S}$ (Cheng, et al., 2017).

i. Fisher Score

The supervised feature selection algorithm called Fisher Score selects features where instances from the same class have small feature values while instances from different classes have larger feature values, assuming they come from c classes with labels $y = \{y_1, y_2, \dots, y_n\}$ among n samples. To compute the Fisher score for each feature (Duda , Hart , & Stork, 2012):

$$fisher.score(f_i) = \frac{\sum_{j=1}^c n_j (\mu_{i,j} - \mu_i)^2}{\sum_{j=1}^c n_j \sigma_{i,j}^2}. \quad (20)$$

Here, n_j , μ_i , $\mu_{i,j}$ and $\sigma_{i,j}^2$ respectively indicate the number of samples in class j , the mean value of feature f_i , the mean value of attribute f_i for samples in class j , and the variance of feature f_i for samples in class j . Like Laplacian Score, the top features can be obtained by selecting attributes with the highest Fisher score (Cheng, et al., 2017).

The Fisher Score can be considered as a different version of the Laplacian score when the similarity (affinity) matrix is as follows (He, Cai, & Niyogi, 2005):

$$S(i, j) = \begin{cases} \frac{1}{n_l} & \text{if } y_i = y_j = l, \\ 0 & \text{otherwise} \end{cases}, \quad (21)$$

where n_l denotes the number of data samples in class l . With this similarity matrix, the relationship between the Fisher Score and the Laplacian Score can be calculated as follows:

$$fisher.score(f_i) = 1 - \frac{1}{laplacian.score(f_i)}. \quad (22)$$

ii. Trace Ratio Criterion (Supervised)

The trace ratio criterion, a different algorithmic method to select a globally optimal subset of features, measures feature importance using a trace ratio norm. It constructs two proximity matrices, S_w for intra-class and S_b for inter-class data similarity expression. The corresponding diagonal matrices and Laplacian matrices are defined as $D_w(i, i) = \sum_{j=1}^n S_w(i, j)$, $D_b(i, i) = \sum_{j=1}^n S_b(i, j)$, $L_w = D_w - S_w$, $L_b =$

$D_b - S_b$. It is represented by a selection indicator matrix $W = [w_{i_1}, w_{i_2}, \dots, w_{i_k}] \in \mathbb{R}^{d \times k}$, where only the i_j^{th} entry in w_{i_j} is 1, and other entries are 0. Along with these, the trace ratio criterion for all k features in F is defined as follows:

$$trace.ratio(F) = \frac{tr(W'X'L_bXW)}{tr(W'X'L_bXW)}. \quad (23)$$

The main objective is to increase data similarity among samples from the same class while reducing data similarity among samples from different classes.

iii. ReliefF (Supervised)

Relief and its multi-class version, ReliefF, are supervised filter algorithms that select features based on their ability to distinguish between different classes of features. Assuming l data samples among all n samples are randomly selected, the feature score f_i is defined as follows (Robnik-Sikonja & Kononenko, 2003):

$$Relief.score(f_i) = \frac{1}{2} \sum_{j=1}^l d(X(j, i) - X(NM(j), i)) - d(X(j, i) - X(NH(j), i)), \quad (24)$$

here, $NM(j)$ and $NH(j)$ indicate the nearest samples with the same class label and a different class label to x_j using the Euclidean distance metric $d(\cdot)$. As Relief is used for binary classification problems, its extended form for multi-class classification, as per Equation (24), has transformed as follows:

$$Relief F.score(f_i) = \frac{1}{c} \sum_{j=1}^l \left(-\frac{1}{m_j} \sum_{x_r \in NH(j)} d(X(j, i) - X(r, i)) + \sum_{y \neq y_j} \frac{1}{h_{jy}} \frac{p(y)}{1-p(y)} \sum_{x_r \in NM(j, y)} d(X(j, i) - X(r, i)) \right), \quad (25)$$

$NH(j)$ and $NM(j, y)$, with dimensions h_{jy} and m_j respectively, denote the nearest samples to x_j within the same class and a different class label y . The ratio of samples with the class label y is represented as $p(y)$.

3.3.1.2 Information theoretical based methods. In information theoretical based feature selection methods, the importance of features is generally determined using various intuitive filter criteria. In the literature, different information-theoretic criteria have been proposed to maximize feature distinctiveness and minimize feature redundancy (Duda, Hart, & Stork, 2012). The suitability of a feature is often measured by its correlation with class labels, and most algorithms in this category are supervised. Additionally, many information-theoretic concepts are applicable only to discrete variables, restricting feature selection algorithms in this subgroup to discrete data. For

numerical feature values, some data discretization methods need to be employed (Dougherty, Kohavi, & Saham, 1995). Studies focusing on information-theoretic criteria are gathered within the framework of conditional likelihood maximization, with many algorithms tailored as a specific instance of this framework (Brown, Pocock, Zhao, & Lujan, 2012). This subsection introduces some representative algorithms. Firstly, entropy, one of the fundamental information-theoretic concepts, measures the uncertainty of a discrete random variable. The entropy of a discrete random variable X is defined as follows:

$$H(X) = - \sum_{x_i \in X} P(x_i) \log(P(x_i)), \quad (26)$$

where x_i represents a specific value of the random variable X , $P(x_i)$ defines the probability that X_i can be predicted from all possible X values in the data. Conditional entropy of X given another discrete random variable Y is expressed as follows:

$$H\left(\frac{X}{Y}\right) = \sum_{y_j \in Y} P(y_j) \sum_{x_i \in X} P(x_i|y_j) \log(P(x_i|y_j)), \quad (27)$$

in which $P(y_j)$ stands for the prior probability of y_j , while $P(x_i|y_j)$ refers to the conditional probability of x_i given y_j . The measure of conditional entropy signifies the uncertainty of X given another discrete random variable Y . The concept of information gain between X and Y is utilized to measure their dependencies on entropy and conditional entropy (Shannon, 1948). As it quantifies the information shared by X and Y together, mutual information is calculated below:

$$I(X; Y) = H(X) - H(X|Y) = \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)}. \quad (28)$$

Here, $P(x_i, y_j)$ denotes the joint probability of x_i and y_j . Information gain is symmetric, meaning that $I(X; Y) = I(Y; X)$, and it is zero if discrete X and Y variables are independent. Similar to the concept of entropy, conditional mutual information (CMI) of a third discrete variable Z given X and Y is defined as follows:

$$I(X; Y|Z) = H(X|Z) - H(X|Y, Z) = \sum_{z_k \in Z} P(z_k) \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i, y_j|z_k) \log \frac{P(x_i, y_j|z_k)}{P(x_i|z_k)P(y_j|z_k)}. \quad (29)$$

When a third discrete variable Z is introduced, it defines the shared mutual information

by X and Y . Finding the global solution to the problem seeking the best set of features is known as an NP-hard problem. Therefore, most algorithms in this family rely on intuitive sequential search approaches to add or remove features individually. In the feature selection problem, forward sequential search selects features one by one to add them into the selected feature set. S denotes the initially empty current selected feature set. Y represents class labels. $X_j \in S$ indicates a specific feature in the current set S . $J(\cdot)$ represents a feature selection criterion, where higher values of $J(X_k)$ denote higher importance of the feature X_k . In the combined conditional likelihood maximization for feature selection, the selection criterion (score) for an unselected new feature X_k is given as follows:

$$J_{cmi}(X_k) = I(X_k; Y) + \sum_{X_j \in S} g[I(X_j; X_k), I(X_j; X_k|Y)], \quad (30)$$

the function $g(\cdot)$ here is a function of two variables, namely $I(X_j; X_k)$ and $I(X_j; X_k|Y)$ and it's a nonlinear function depending on these values. These two variables are defined as a criterion based on the linear combination of Shannon information terms:

$$J_{cMI}(X_k) = I(X_k; Y) - \beta \sum_{X_j \in S} I(X_j; X_k) + \lambda \sum_{X_j \in S} I(X_j; X_k|Y). \quad (31)$$

Here, β and λ represent two non-negative parameters between 0 and 1, and if $g(\cdot)$ is a linear function of these two variables, then this criterion is referred to as a measure based on the nonlinear combination of these two variables, representing Shannon information terms (Shannon, 1948).

i. MIM-Mutual Information Maximization (or Information Gain)

The method of measuring the importance of a feature by its correlation with the class label is known as Mutual Information Maximization (MIM). If a feature has a strong correlation with the class label, it tends to provide good classification performance. For a newly unselected feature X_k , the Mutual Information score is calculated as follows (Lewis, 1992):

$$J_{MIM}(X_k) = I(X_k; Y). \quad (32)$$

In this method, the feature score is independent of other features, focusing solely on feature correlation while disregarding feature redundancy. Once the MIM feature scores for all unselected features are obtained, the feature with the highest score is selected and added to the selected feature set. This process continues until the desired

number of features is achieved. It can be observed that MIM is a specific case of a linear combination of Shannon information terms:

$$J_{CMI}(X_k) = I(X_k; Y) - \beta \sum_{X_j \in S} I(X_j; X_k) + \lambda \sum_{X_j \in S} I(X_j; X_k | Y), \quad (33)$$

where β and λ are equal to zero.

ii. MIFS - Mutual Information Feature Selection

MIM's drawback lies in assuming feature independence. Powerful features shouldn't merely correlate strongly with class labels; they should also possess minimal correlation among themselves. The Mutual Information Feature Selection (MIFS) criterion calculates a feature's score for a new unselected feature during the selection phase by considering both the significance of the feature and the redundancy among features (Battiti, 1994):

$$J_{MIFS}(X_k) = I(X_k; Y) - \beta \sum_{X_j \in S} I(X_k; X_j). \quad (34)$$

The relevance of the new feature is evaluated through the first term, $I(X_k; Y)$, while the second term aims to confirm the feature with high mutual information among selected features, thereby minimizing feature redundancy. MIFS can also be considered a specific case of Shannon's linear combination under certain information conditions:

$$J_{CMI}(X_k) = I(X_k; Y) - \beta \sum_{X_j \in S} I(X_j; X_k) + \lambda \sum_{X_j \in S} I(X_j; X_k | Y), \quad (35)$$

where β is set between 0 and 1, λ is set to 0.

iii. MRMR-Minimum Redundancy Maximum Relevance

In this method, unlike MIFS which can take the β value as 1, Minimum Redundancy Maximum Relevance (MRMR) adopts a criterion that sets the β value inversely proportional to the number of selected features (Peng, Long, & Dinc, 2005):

$$J_{MRMR}(X_k) = I(X_k; Y) - \frac{1}{|S|} \sum_{X_j \in S} I(X_k; X_j). \quad (36)$$

The impact of redundancy of features is gradually decreases by selecting more features. When selecting additional non-redundant features, it becomes tougher for new ones to outweigh the relevance of those already in set S . There might be an interpretation that adding more features to S strengthens the pairwise independence among them, likely due to data noise (Brown, Pocock, Zhao, & Lujan, 2012). The

MRMR criterion relies on maximizing conditional mutual information:

$$J_{CMI}(X_k) = I(X_k; Y) - \beta \sum_{X_j \in S} I(X_j; X_k) + \lambda \sum_{X_j \in S} I(X_j; X_k | Y), \quad (37)$$

Here, the β can be changed with $\frac{1}{|S|}$ and λ is set to zero.

iv. CIFE - Conditional Infomax Feature Extraction

MIFS and MRMR are methods that consider both feature relevance and feature redundancy. Moreover, literature suggests that, rather than solely minimizing feature redundancy, there should also be a maximization of conditional redundancy between the unselected features and the class labels given the previously selected features (Lin & Tang, 2006; Akadi, Ouardighi, & Aboutajdine, 2008; Guo & Nixon, 2009). If the conditional redundancy between class labels and feature redundancy is stronger than redundancy among features, it adversely affects feature selection. Therefore, feature selection is computed based on the Conditional Infomax Feature Extraction (CIFE) criterion, which calculates the feature score for a new unselected feature, X_k :

$$J_{CIFE}(X_k) = I(X_k; Y) - \sum_{X_j \in S} I(X_j; X_k) + \sum_{X_j \in S} I(X_j; X_k | Y). \quad (38)$$

Compared to MIFS, it is observed that in order to maximize conditional redundancy, a third criterion, $\sum_{X_j \in S} I(X_j; X_k | Y)$ has been added. Additionally, CIFE is also a specific instance of the linear combination of Shannon information terms.

$$J_{CMI}(X_k) = I(X_k; Y) - \beta \sum_{X_j \in S} I(X_j; X_k) + \lambda \sum_{X_j \in S} I(X_j; X_k | Y), \quad (39)$$

where λ and β values are equal to 1.

v. JMI- Joint Mutual Information

MIFS and MRMR reduce feature redundancy in the feature selection process. An alternative criterion, Joint Mutual Information, is suggested to increase the shared complementary information between a newly unselected feature and the class labels given the selected features (Yang & Moody, 1999). The feature selection criterion is listed as follows (Meyer, Schretter, & Bontempi, 2008):

$$J_{CMI}(X_k) = \sum_{X_j \in S} I(X_k, X_j; Y). \quad (40)$$

The fundamental idea behind JMI is that we should include new features that complement the existing features given the class labels. Unlike previous approaches that can be directly represented by a linear combination of Shannon information terms,

JMI cannot be directly reduced to probability maximization framework. Studies have shown that the JMI criterion can be rewritten as follows with simple manipulations (Duda , Hart , & Stork, 2012):

$$J_{JMI}(X_k) = I(X_k; Y) - \frac{1}{|S|} \sum_{X_j \in S} I(X_j; X_k) + \frac{1}{|S|} \sum_{X_j \in S} I(X_j; X_k | Y). \quad (41)$$

where β and λ is $\frac{1}{|S|}$.

vi. CMIM-Conditional Mutual Information Maximization

The mentioned information-theoretic feature selection criteria can be expressed as a linear combination of Shannon information terms. Besides these, Conditional Mutual Information Maximization (CMIM) can be reduced to a nonlinear combination of Shannon information. CMIM is a recurring criterion that selects features maximizing mutual information with given class labels, even if it has previously selected features (Vidal-Naquet & Ullman, 2003; Fleuret, 2004). Despite its strength in maximizing mutual information for class labels, the CMIM method does not select features similar to previously chosen features. During the selection process, the feature score for each new unselected feature X_k is expressed as follows:

$$J_{CMIM}(X_k) = \min_{X_j \in S} [I(X_k; Y | X_j)]. \quad (42)$$

If X_k is not highly related to the class label Y or X_k is redundant given S , it should not be overlooked that the value of $I(X_k; Y | X_j)$ is small. By selecting the feature that maximizes this minimum value, it ensures strong predictive capabilities of the selected features and guarantees the reduction of feature redundancy. After some derivatives, the CMIM criterion has been calculated as follows:

$$J_{CMIM}(X_k) = I(X_k; Y) - \max_{X_j \in S} [I(X_j; X_k) - I(X_j; X_k | Y)]. \quad (43)$$

Therefore, CMIM has also been defined as a special case of conditional mutual information maximization:

$$J_{cmi}(X_k) = I(X_k; Y) + \sum_{X_j \in S} g[I(X_j; X_k), I(X_j; X_k | Y)]. \quad (44)$$

vii. DISR-Double Input Symmetrical Relevance

In another class of information-theoretical-based methods such as Dual Input Symmetric Relevance (DISR), normalization methods have been utilized for normalizing mutual information (Guyon, Nikravesh, Gunn, & Zadeh, 2006; Meyer, Schretter, & Bontempi, 2008).

$$J_{DISR}(X_k) = \sum_{X_j \in S} \frac{I(X_j X_k; Y)}{H(X_j X_k Y)}. \quad (45)$$

It has been stated that DISR is a non-linear combination of Shannon information terms and is described as maximizing conditional likelihood (Meyer, Schretter, & Bontempi, 2008).

viii. FCBF-Fast Correlation Based Filter

A different information-theoretical-based feature selection method, Fast Correlation-Based Filter (FCBF) is a filter technique that utilizes both feature-class correlation and feature-feature relationships simultaneously and cannot be expressed as a combined conditional likelihood maximization method (Yu & Liu, 2003). When a predefined threshold is given, it selects a subset of S features that have a high level of correlation with the class label Y , given that SU represents the symmetric uncertainty and $SU \geq \delta$. The SU between a set of features X_S and the class label Y is defined as follows:

$$SU(X_S, Y) = 2 \frac{I(X_S, Y)}{H(X_S) + H(Y)}. \quad (46)$$

i. Interaction Capping-ICAP

Interaction Capping is a similar feature selection criterion to CMIM in Equation (42). The difference lies in the fact that the interaction boundary term $I(X_j; X_k) - I(X_j; X_k|Y)$ cannot be negative (Jakulin, 2005).

$$J_{CMIM}(X_k) = I(X_k; Y) - \sum_{X_j \in S} \max[0, I(X_j; X_k) - I(X_j; X_k|Y)]. \quad (47)$$

When the $g(.)$ function is set to $\max[0, I(X_j; X_k) - I(X_j; X_k|Y)]$, it becomes a nonlinear combination of Shannon information.

3.3.1.3 Sparse learning based methods. Sparse-based feature selection techniques focus on identifying a subset of relevant attributes while inducing sparsity within the feature space. These methods aim to select a compact set of features by encouraging a sparse representation, where only a few attributes are deemed significant for prediction or classification tasks. Lasso regression, a prominent sparse-based technique, imposes a penalty on the absolute magnitude of regression coefficients, effectively driving some coefficients to zero and selecting a subset of features with non-zero coefficients. This property makes Lasso regression particularly adept at feature selection by promoting a sparse solution, allowing it to handle high-

dimensional data while emphasizing the most informative attributes.

Another approach within sparse-based techniques involves sparse coding or dictionary learning, where the data is represented as a linear combination of a few learned dictionary elements. Techniques like Orthogonal Matching Pursuit (OMP) or basis pursuit aim to find a sparse representation of data by iteratively selecting relevant dictionary elements to approximate the input. These methods have found application in signal processing and image compression, where identifying a concise representation of data with minimal information loss is crucial. Sparse coding-based techniques facilitate feature selection by extracting a sparse set of dictionary elements that best represent the data, effectively identifying the most informative attributes.

Sparse-based feature selection techniques offer a unique advantage in handling high-dimensional datasets by promoting simplicity and interpretability through sparse solutions. By encouraging a compact representation with a limited set of influential attributes, these methods mitigate the risk of overfitting and reduce computational complexity, enhancing model efficiency. However, selecting an appropriate sparsity-inducing mechanism and tuning the associated parameters pose challenges, demanding careful consideration of dataset characteristics and trade-offs between sparsity and predictive performance. Nonetheless, sparse-based techniques stand as valuable tools for dimensionality reduction and feature selection, particularly in scenarios where identifying a concise yet effective subset of attributes is vital.

One commonly used criterion for unsupervised feature selection involves selecting features that best preserve the manifold structure of the data (He, Cai, & Niyogi, 2005). Alternatively, encoded discriminative information proven to be effective in various learning tasks is also utilized (Fukunaga, 1990). To discover prominent features using both discriminative information and feature correlations, an $l_{2,1}$ -norm based feature selection algorithm (RFS) has been proposed (Nie, Huang, Cai, & Ding, 2010):

$$\min_{W'W=I_d} \text{tr}(W'MW) + \alpha \|W\|_{2,1}, \quad (48)$$

where $M = X'[\sum_{i=1}^n (P_i H_{k+1} (\bar{X}_i \bar{X}_i' + \lambda I_{k+1})^{-1} H_{k+1} S_i')]$.

3.3.1.4 Statistical based methods. Statistical-based feature selection methods operate on statistical principles to identify the most relevant attributes within a dataset, aiming to enhance model performance and interpretability. These methods employ

statistical metrics, tests, or algorithms to quantify the relationship between features and the target variable, as well as to assess the significance and relevance of individual attributes. An array of statistical techniques is employed, ranging from univariate methods that evaluate features independently to multivariate approaches that consider interactions among attributes.

Univariate statistical methods, such as t-tests, ANOVA (analysis of variance), or chi-square tests, assess the relationship between each feature and the target variable in isolation. These methods assign statistical scores or p-values to features, ranking them based on their individual predictive power or contribution to the target variable. Conversely, multivariate statistical techniques, like principal component analysis (PCA) or linear discriminant analysis (LDA), consider interactions among features to extract a subset of attributes that collectively explain the variance or discriminative power within the data. These methods reduce dimensionality by transforming features into a lower-dimensional space while retaining as much relevant information as possible.

Statistical-based feature selection methods offer a systematic approach to identify informative attributes and reduce data dimensionality while accounting for the statistical significance of individual features. By leveraging statistical tests and metrics, these methods enable the selection of attributes that contribute most significantly to predictive accuracy or class separability, thereby enhancing model efficiency and interpretability. However, the efficacy of these methods relies on assumptions about data distributions, linearity, and independence among features, necessitating careful consideration of the dataset characteristics and the applicability of statistical techniques to the specific learning task at hand. The majority of statistical-based algorithms evaluate features individually, hence overlooking feature redundancy during the selection phase. It's worth noting that a significant portion of algorithms in this category operates with discrete data (Li, et al., 2018)

i. F-Score (Supervised)

When both within-class variance and between-class variance are evaluated, the f -score of a feature can be calculated as follows (Wright, 1965):

$$f_{score}(f_i) = \frac{\sum_j \frac{n_j}{c-1} (\mu_j - \mu)^2}{\frac{1}{n-c} \sum_j (n_j - 1) \sigma_j^2}. \quad (49)$$

Where the given feature f_i , n_j , μ , μ_j and σ_j represent respectively the number of

samples in class j , the mean feature value, the mean feature value in class j , and the standard deviation of the feature value in class j . Similar to the t -score, the f -score and feature importance are directly proportional to each other.

ii. Gini Index (Supervised)

The Gini index is a statistical measure used to assess the similarity of features from different classes. For a feature f_i , the Gini index score is given as follows (Gini, 1913):

$$gini.index.score(f_i) = \min_w(p(W)(1 - \sum_{s=1}^c p(C_s|W)^2 + p(\bar{W})(1 - \sum_{s=1}^c p(C_s|\bar{W})^2)), \quad (50)$$

where C_s represents the class label s , and $p(\cdot)$ denotes probability. The Gini index is obtained by traversing through all possible splits W . Typically, in binary classification problems, it can take a maximum value of 0.5 and can also be used in multi-classification problems. In contrast to previous statistical measurements, the feature importance is inversely proportional to the Gini index value (Li, et al., 2018).

iii. CFS (Supervised)

The core idea behind the Correlation-based Feature Selection (CFS), an intuitive heuristic method, is to calculate the merit of a feature subset F (Hall & Smith, 1999):

$$CFS.score(F) = \frac{k\bar{r}_{cf}}{\sqrt{k + k(k-1)\bar{r}_{ff}}}, \quad (51)$$

here, the CFS score represents the intuitive “merit” of a subset of k features in F . r_{cf} denotes the average feature-class correlation, while r_{ff} indicates the average feature-feature inter-correlation. In the Equation (51) above, the numerator demonstrates the estimated strength of the selected features, while the denominator indicates how much redundancy exists within the feature set. In this method, a good feature subset should have strong correlations with class labels and weak correlations among themselves. To obtain the feature-class and feature-feature correlations, CFS utilizes symmetric uncertainty to estimate the relationship between two features (Li, et al., 2018).

iv. Chi-Square (Supervised)

This method can be used to select n features from X with the highest Chi-Square statistics values, but it is only feasible when the classes contain Boolean or non-negative features. In statistics, the Chi-Square test measures the dependency between stochastic variables. Therefore, the aim of this method is to separate features that are most likely independent of the class, hence deemed irrelevant for classification

purposes.

3.3.1.5 Wrapper methods. Wrapper methods in feature selection represent a class of techniques that embed the feature selection process within the model building itself. Unlike filter methods that assess features independently of the learning algorithm, wrapper methods leverage specific models to evaluate subsets of features. These methods typically utilize a search strategy, often heuristic-based, to traverse the feature space, evaluating subsets' performance using a chosen machine learning algorithm. Through an iterative process, wrapper methods aim to find the most relevant subset of features that optimize the model's performance based on a defined evaluation criterion, such as accuracy or error rate. Examples of wrapper methods include forward selection, backward elimination, and recursive feature elimination. Despite their computational intensity due to exhaustive search strategies, wrapper methods tend to yield more accurate results as they directly incorporate the model's performance feedback into the feature selection process, making them especially effective for complex datasets and diverse machine learning models.

One of the key advantages of wrapper methods lies in their ability to capture the interaction between features and the learning algorithm's behavior, allowing for a more tailored and precise selection of attributes. By integrating the model's performance as the guiding criterion for feature selection, wrapper methods can potentially uncover intricate relationships and dependencies between features that might be missed by filter or embedded methods. However, their reliance on specific learning algorithms and exhaustive search procedures can make them computationally demanding, especially for high-dimensional datasets. The trade-off between computational resources and the potential for higher accuracy underscores the nuanced balance in choosing wrapper methods, emphasizing their significance in scenarios where optimizing model performance through feature selection is paramount.

i. Sequential Backward Selection Algorithm Based on Support Vector Machine and Decision Tree

Another effective feature selection algorithm is the Sequential Backward Selection (SBS) method (Marill & Green, 1963). SBS aims to find the best subset of features that minimizes the number of features from the entire dataset while maximizing accuracy measured by Support Vector Machine (SVM) and Decision Tree (DT) methods. In contrast to the Sequential Forward Selection (SFS) algorithm, this method starts with all features and removes one feature at each step until there is no

improvement in accuracy. Once removed, features cannot be added back to the subset during the selection process.

A multitude of techniques and methodologies exist within the feature selection, catering to diverse data types, model complexities, and optimization objectives. Techniques like genetic algorithms, recursive feature elimination, feature importance ranking via ensemble methods (e.g., Random Forest, Gradient Boosting), and sparse models (e.g., Lasso, Elastic Net) provide further avenues for feature selection, each with its own strengths and applicability based on the nature and requirements of the data and the learning task.

Overall, the choice of feature selection technique depends on various factors, including the dataset size, dimensionality, computational resources, and the specific goals of the machine learning task. Selecting an appropriate feature selection method is crucial in ensuring model robustness, interpretability, and generalization to unseen data.

The total of 19 feature selection algorithms mentioned in the literature has been implemented by using Python and compared with the proposed pruning technique which is given in Equation (18). The suggested pruning method has been described in Chapter 3.2 and its results have been shared in Section 4.1.

3.3.2 Feature selection via second-order cone programming. The method which is modeled in Section 3.2 with Equation (18) by using second-order cone programming to prune the ensemble of CNNs, has been adapted for the feature selection problem in this thesis. All features in the dataset were trained in a simple neural network, followed by the generation of ensemble from each feature. The features are pruned by using second-order cone programming optimization model and the performance of the pruned model was compared with other feature selection techniques in the literature.

$$\begin{aligned}
\min \quad & (\alpha)(t) + (1 - \alpha)(c^T x) + \lambda u \\
\text{st.} \quad & \begin{bmatrix} 1 + t \\ 2Lx \\ 1 - t \end{bmatrix} \geq 0 \\
& \sum_{i=1}^n |x_i| \leq u \\
& x \in K
\end{aligned} \tag{18}$$

where $K = K^{n_1} \dots K^{n_r} R^{n_i}$ when K^{n_i} is either a second-order cone or a rotated second-order cone $K_r^{n_i}$ while $(x_i) \in K_q^2$.

The proposed second-order cone programming model prunes trained features in the artificial neural network. Each selected feature is retrained and tested in the same neural network. For this purpose, five different datasets were used, and information regarding the datasets is provided in Table 1.

Table 1

Dataset and Their Definitions

Dataset	Feature Type	Number of Features	Number of Instances	Number of Classes
Lungsmall	Bio	325	73	7
Glioma	Bio	4434	50	4
Yale	Image	1024	165	15
Basehock	Text	4862	1993	2
Isolet	Real	617	7797	5

The selected datasets were chosen randomly, commonly used in feature selection problems (Li, et al., 2018). The proposed method is independent of the type of problem the data originates from. All algorithms recommended in this project across all work packages are generalized models independent of the data source. Randomly chosen datasets from the literature are used for testing the algorithm performance, with no specific reason behind their selection. As shown in Table 1, Lung small dataset consist of 325 features, 73 instances with 7 classes which belongs to healthcare domain. Glioma dataset which also belongs to healthcare domain consists of 4434 features, 50 instances with 4 classes. Yale dataset keeps images of individuals with 1024 features, 165 instances and 15 classes. Basehock dataset has 4862 features, 1993 instances and 2 classes where Isolet has 617 features, 7797 instances and 5 classes. As shown in Table 1, 30% of the datasets were allocated for testing, while the remaining 70% were used as training data. Preprocessing was conducted for each dataset, involving normalization specifically on the portion allocated for training. The success rates were compared using the features selected by different feature selection methods through 5-fold cross-validation. The flowchart of the proposed method is illustrated in Figure 6 and the pseudocode of the FSOCP method is given in Algorithm 2.

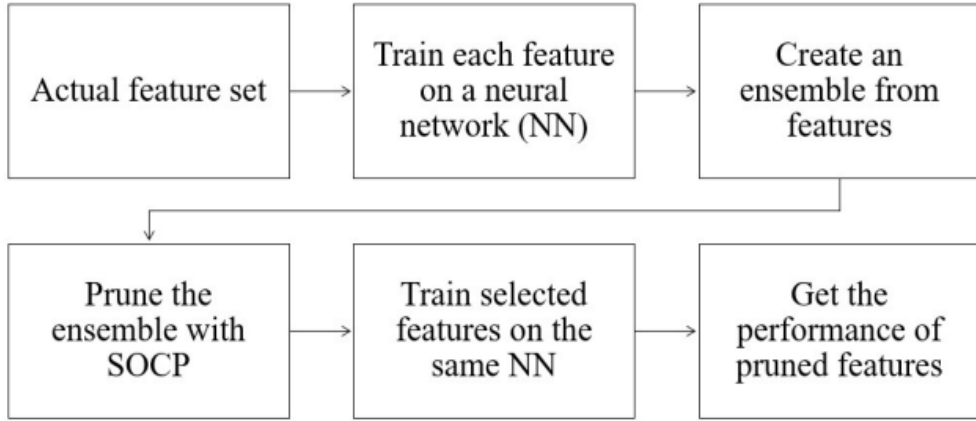


Figure 6 The flowchart of the FSOCP algorithm

Algorithm 2 : *Pseudocode of the FSOCP algorithm*

Input: D : dataset, F : Number of features of a dataset

Parameters: n_m : A random sequential neural network model, l : λ values, a : α values, e : experiment size which is $\lambda\alpha$, t : threshold for pruning

Output: P : accuracy of pruned ensemble, B_r : best experiment, A_r : average experiment, W_r : worst experiment

for $i \leftarrow 1$ **to** F **do**

 Split D for training $D_{training}$ and testing $D_{testing}$

 Train with Adam Optimizer & Categorical Cross Entropy Loss on n

 Generate the E where E = Ensemble from features.

end

Select λ and α values randomly, (In this thesis; $\alpha = (0,1;0,2;0,3;0,4;0,5)$ and for $\lambda = (0,1;0,3;0,5;0,7;0,9)$)

Solve SOCP and get $w_{E,e}$ where w = weights of features

Prune w according to t

Get k where k = number of selected features

Train and Test N on m

Get P , B_r , A_r and W_r .

Feature selection allows for the identification of the most important features, thereby enhancing the effectiveness, productivity, and intelligibility of decision-making models. By pinpointing the most informative features, these methods facilitate more precise evaluations, deeper insights, and well-informed decisions across various domains. Despite its inherent challenges,

feature selection has emerged as a dynamic area of research owing to its profound relevance in numerous fields. It's imperative to meticulously assess the chosen approach to ensure the resulting model achieves optimal performance and interpretability. Consequently, this study introduces a novel feature selection technique termed Feature Selection via Second Order Cone Programming (FSOCP), founded on ensemble pruning using second order cone programming and neural networks. Rigorous performance comparisons of FSOCP against various feature selection methods across diverse datasets illustrate its viability, yielding favorable outcomes. FSOCP notably exhibits superior efficiency, especially in terms of algorithmic training and overall processing time. Unlike prevailing methods that rely on exhaustive brute-force strategies for feature subset selection, FSOCP's optimization-based framework offers a transparent mathematical explanation, enhancing the comprehensibility of the algorithm's underlying mechanisms.

Chapter 4

Results

This chapter of the thesis is divided in two subgroups where the first part gives results and findings on pruning the ensemble of CNNs with employing SOCP and the second part shifts the focus to the findings derived from using the proposed model as an embedded feature selection algorithm which is FSOCP.

4.1 Results of Pruning the Ensemble of CNNs with SOCP

In the Equation (18), α and λ values are hyperparameters tuned through cross-validation. The chosen range for these parameters per dataset was $\alpha = (0.1, 0.2, 0.3, 0.4, 0.5)$ and $\lambda = (0.1, 0.3, 0.5, 0.7, 0.9)$. The Equation (18) provides weight values for each model in the ensemble by solving SOCP. Sparse weights guide the selection of CNN models for the sub ensemble. Voting technique is used among these models to determine the final classification. Different thresholds are experimented on the training set for weights close to “0” in sparse solution of the Equation (18). Voting aggregates predictions from selected models to yield final test scores. These results can be seen below in Table 2 and Table 3. Table 2 presents prediction values, model count, and threshold values for pruned and full ensembles using voting. Table 3 shows max, average, and min accuracy estimations for pruned and full ensemble models without voting (Güldoğuş, Abdullah, Ali, & Akyüz, 2023).

Table 2

Performance Result of Full and Pruned Ensemble with SOCP (Güldoğuş, Abdullah, Ali, & Akyüz 2023)

	Full Ensemble	Pruned Ensemble
CIFAR-10		
Test Accuracy	85.05%	90.09%
Number of Models	50	38
Threshold	n/a	0.131
Test Accuracy	85.79%	90.57%
Number of Models	150	96
Threshold	n/a	0.131
Test Accuracy	90.33%	90.12%
Number of Models	300	56
Threshold	n/a	0.131
CIFAR-100		
Test Accuracy	53.75%	64.80%
Number of Models	50	48

Table 2 (Cont.)		
Threshold	n/a	0.188
Test Accuracy	52.92%	65.70%
Number of Models	150	98
Threshold	n/a	0.188
Test Accuracy	59.54%	59.28%
Number of Models	300	98
Threshold	n/a	0.188
MNIST		
Test Accuracy	98.24%	99.59%
Number of Models	50	37
Threshold	n/a	0.18
Test Accuracy	98.12%	99.68%
Number of Models	150	94
Threshold	n/a	0.18
Test Accuracy	99.66%	99.64%
Number of Models	300	170
Threshold	n/a	0.18

Pruned models are 56 for CIFAR-10, 98 for CIFAR-100, and 170 for MNIST) closely match the performance of full models 300. Pruned models offer advantages in terms of time and complexity compared to unpruned ones.

Table 3

Best, average and worst performance results of CIFAR-10, CIFAR-100 and MNIST

	Full Ensemble	Pruned Ensemble
CIFAR-10		
Max. Accuracy Score	85.05%	90.09%
Avg. Accuracy Score	80.48%	89.98%
Min. Accuracy Score	71.56%	89.91%
Number of Models	50	38
Max. Accuracy Score	85.79%	90.57%
Avg. Accuracy Score	80.67%	90.52%
Min. Accuracy Score	69.40%	90.50%
Number of Models	150	96
Max. Accuracy Score	86.11%	84.44%
Avg. Accuracy Score	81.28%	79.10%
Min. Accuracy Score	69.23%	69.23%
Number of Models	300	56
CIFAR-100		
Max. Accuracy Score	53.57%	64.79%
Avg. Accuracy Score	42.50%	64.60%
Min. Accuracy Score	28.91%	64.31%
Number of Models	50	48

Table 3 (Cont.)		
Max. Accuracy Score	52.92%	65.73%
Avg. Accuracy Score	42.64%	65.70%
Min. Accuracy Score	24.44%	65.68%
Number of Models	150	95
Max. Accuracy Score	49.45%	49.45%
Avg. Accuracy Score	33.36%	34.08%
Min. Accuracy Score	11.52%	11.52%
Number of Models	300	98
MNIST		
Max. Accuracy Score	98.24%	99.59%
Avg. Accuracy Score	98.12%	99.47%
Min. Accuracy Score	97.67%	99.33%
Number of Models	50	37
Max. Accuracy Score	98.12%	99.68%
Avg. Accuracy Score	99.28%	99.57%
Min. Accuracy Score	97.45%	99.43%
Number of Models	150	94
Max. Accuracy Score	99.66%	99.64%
Avg. Accuracy Score	99.24%	99.23%
Min. Accuracy Score	95.22%	97.86%
Number of Models	300	170

This thesis explores ensemble learning combined with pruning techniques to enhance image classification tasks using CIFAR-10, CIFAR-100, and MNIST datasets. Creating ensembles of CNNs (50, 150, 300 models) and pruning them with SOCP improves performance without relying on transfer learning or pre-trained models, focusing on achieving superior results with fewer models.

4.2 Results of FSOCP

The performance of the FSOCP feature selection technique proposed here compared with 19 different methods found in the literature. Table 4 demonstrates that FSOCP stands out as a faster alternative with better performance, making it a promising choice for feature selection. Results in the table showcase the FSOCP method's superior performance, particularly noted for Glioma, Lung and Basehock datasets, surpassing the methods cited in the literature. While highest average accuracy is 0.87 for CFS, Gini Index, ICAP, and Trace Ratio algorithms, FSOCP remains competitive and could be tested alongside these methods to confirm its effectiveness. Moreover, the proposed FSOCP model outpaces other methods in terms of training and total processing time, showcasing its efficiency.

Table 4.

Performances of the proposed FSOCP algorithm and other feature selection techniques in the literature

Methods/Dataset	Accuracy Values						
	Glioma	Isolet	Lung	Yale	Basehock	Avg.	Std.
CFS	0.80	0.95	0.88	0.83	0.93	0.87	0.05
Chi-Square	0.73	0.95	0.87	0.80	0.82	0.83	0.07
CIFE	0.78	0.95	0.86	0.79	0.93	0.86	0.06
CMIM	0.70	0.95	0.89	0.84	0.94	0.86	0.09
DecisionTree	0.73	0.95	0.90	0.80	0.94	0.86	0.08
DISR	0.73	0.95	0.88	0.81	0.93	0.86	0.08
F-Score	0.67	0.94	0.85	0.82	0.88	0.83	0.09
FCBF	0.73	0.95	0.89	0.83	0.93	0.86	0.07
Fisher Score	0.80	0.95	0.84	0.80	0.93	0.86	0.06
GINI Index	0.76	0.95	0.89	0.86	0.93	0.87	0.06
ICAP	0.76	0.95	0.87	0.82	0.95	0.87	0.07
JMI	0.68	0.95	0.90	0.80	0.93	0.85	0.1
MIFS	0.67	0.96	0.87	0.82	0.94	0.85	0.1
MIM	0.75	0.95	0.83	0.84	0.94	0.86	0.07
MRMR	0.73	0.95	0.90	0.80	0.94	0.86	0.08
ReliefF	0.64	0.95	0.90	0.82	0.93	0.84	0.11
RFS	0.71	0.94	0.88	0.81	0.93	0.85	0.06
SVM	0.75	0.95	0.81	0.82	0.94	0.85	0.07
Trace Ratio	0.76	0.95	0.87	0.85	0.92	0.87	0.06

Table 4 (Cont.)							
FSCOP- % 70 Training	0.80	0.86	0.91	0.60	0.98	0.83	0.12
FSOCP - %60 Training	0.80	0.84	0.87	0.41	0.99	0.78	0.19
FSOCP - %80 Training	0.90	0.70	0.94	0.58	0.99	0.82	0.15

All algorithms used in this project were implemented in *Python*, utilizing libraries like *sklearn*, *numpy*, *scipy*, *cvxpy*, and others. The running time of the proposed FSOCP algorithm aligns with an average feature selection method. Specific running times for each dataset are shared in Table 5. In contrast, the FSOCP model proposed by Equation (18) optimizes feature subsets, taking an average of 1 hours 44 minutes regarding these datasets. This significant time reduction especially on Basehock with largest feature number comes with closely comparable success rates, as demonstrated in Table 4.

Table 5.

Running times of feature selection algorithms

Methods/Dataset	Running Times				
	Glioma	Isolet	Lung	Yale	Basehock
CFS	0:42:13	7:39:09	1:52:45	0:46:42	1 day 23:55:01
Chi-Square	0:00:02	0:00:19	0:00:01	0:00:03	0:00:01
CIFE	2:31:15	1:06:24	0:00:50	0:20:09	2 days 9:35:28
CMIM	2:23:52	1:06:23	0:00:52	0:20:47	2 days 10:17:23
DecisionTree	0:00:02	0:00:19	0:00:00.1	0:00:03	0:00:01
DISR	6:00:55	3:06:47	0:01:59	0:56:05	7 days 2:14:43
F-Score	0:00:02	0:00:19	0:00:00.09	0:00:03	0:00:02
FCBF	0:00:09	0:00:42	0:00:08	0:00:07	0:20:12
Fisher Score	0:00:13	0:00:22	0:00:00.3	0:00:04	0:00:34

Table 5 (Cont.)					
GINI Index	0:01:28	0:26:43	0:00:00.9	0:02:01	0:00:24
ICAP	2:19:03	1:08:35	0:00:43	0:20:31	2 days 10:03:14
JMI	2:18:01	1:04:46	0:00:46	0:20:21	2 days 8:47:56
MIFS	2:22:40	1:06:19	0:00:43	0:20:19	2 days 9:33:21
MIM	2:18:33	1:05:22	0:00:50	0:20:03	2 days 8:55:34
MRMR	2:22:29	1:11:19	0:00:44	0:20:23	2 days 9:45:17
ReliefF	0:00:11	0:01:52	0:00:00.3	0:00:04	0:01:28
RFS	0:18:18	0:03:04	0:00:02	0:00:29	3:52:25
SVM	0:00:02	0:00:20	0:00:00.08	0:00:03	0:00:01
Trace Ratio	0:00:06	0:00:22	0:00:00.16	0:00:03	0:00:25
FSCOP - %70 Training	0:59:02	0:38:29	0:02:29	0:11:07	6:11:10
FSOCP - %60 Training	0:49:19	0:32:51	0:02:15	0:10:31	5:24:32
FSOCP - %80 Training	0:55:55	0:42:04	0:02:17	0:12:33	6:53:47

Furthermore, the FSOCP model's optimization-based nature provides a mathematical foundation, contrasting with brute-force approaches that exhaustively explore all possibilities. Overall, the FSOCP feature selection model outperforms other methods concerning training and total processing time, offering efficiency and competitive success rates.

Chapter 5

Discussions

In the discussion chapter of this thesis, the focus gravitates toward the proposed method's performance and its pivotal significance in the realm of feature selection. The first part delves into the efficacy of the proposed optimization method to prune the ensemble of CNNs, elucidating its performance on various datasets. By scrutinizing its accuracy, computational efficiency, and robustness across diverse datasets, a comprehensive understanding emerges regarding its practical applicability. This evaluation not only underscores the immediate relevance of the proposed method but also unveils its potential for addressing contemporary challenges in deep learning. Furthermore, this chapter aims to shed light on the method's scalability, adaptability to varying data complexities, and its potential enhancements. As an integral part of this exploration, the discussion forecasts future trajectories for refining the proposed method, potentially harnessing advancements in computational frameworks.

The second segment of this discussion delineates the proposed method's role as an embedded feature selection algorithm and its consequential impact on model development. It accentuates the method's influence on refining model interpretability, enhancing predictive accuracy, and curtailing overfitting or data redundancy. Evaluating its efficacy across multiple domains and diverse applications, this chapter elucidates the tangible benefits accrued from employing the proposed method. Furthermore, the discussion extrapolates on the method's potential in shaping the trajectory of future research in feature selection. This includes considerations for extending its utility to specific domains, optimizing its applicability across different machine learning paradigms, and exploring avenues for seamlessly integrating it into cutting-edge models and analytical frameworks. By examining its adaptability to emerging data-driven challenges and its alignment with evolving research paradigms, this segment sets the stage for envisioning the future landscape of feature selection methodologies.

5.1 Discussions and Future Outlook for Pruning the Ensemble of CNNs with SOCP

This research, firstly, aims to create a mathematical model that prunes CNN ensembles using sparse SOCP, addressing a gap in the existing literature. The strength lies in enhancing performance and simplifying CNN ensembles through pruning.

Modeling the pruning algorithm involves considering the accuracy performance of networks generated at different depths, aiming for diverse networks with varying accuracy rates. However, balancing diversity and accuracy poses a trade-off; increasing accuracy often compromises diversity. The proposed pruning algorithm optimizes this trade-off, remaining independent of dataset domains.

The proposed algorithm is versatile, applicable across datasets, and its success rates are compared against full ensemble models across diverse datasets. The experiments showed approximately a 2% variance, demonstrating the system's robustness. Despite a significant reduction in the model count, similar or better accuracy scores were achieved across all datasets.

Recent literature which is given in Section 2.1 indicates that pretrained models and transfer learning improve model performance, especially in deep learning. However, training large CNN ensembles on datasets like ImageNet with transfer learning presents limitations. Transfer learning heavily relies on large, relevant datasets, and without suitable fine-tuning data, its benefits diminish. Overfitting and difficulty adapting to specific tasks or domains can occur when transferring knowledge from pretrained models. Fine-tuning also demands extensive hyperparameter tuning and domain-specific adjustments.

Moreover, pretrained models may carry biases or patterns irrelevant to the target dataset, impacting performance. Additionally, these models often possess large sizes and high computational requirements, limiting their deployment on resource-constrained platforms.

However, experimental results from this study showcase that the proposed pruned ensemble approach, without using transfer learning or pretrained models, achieves competitive or superior performance on CIFAR-10, CIFAR-100, and MNIST datasets. While full ensemble accuracy might lag behind methods employing transfer learning, this approach offers reduced computational complexity and resource needs. By optimizing CNNs ensembles through pruning, it demonstrates superior performance in image classification with fewer models, providing an efficient and effective path for deep learning solutions (Güldoğuş, Abdullah, Ali, & Akyüz, 2023).

5.2. Discussions and Future Outlook on Feature Selection by using SOCP

Feature selection is a crucial problem in machine learning and data mining as it aims to remove redundant or irrelevant attributes from the data to improve prediction accuracy and reduce computation time. Although the concept of ensemble is common,

it is not widely applied to feature selection problems due to the increase in search space and computational costs. Ensemble-based feature selection algorithms offer promising methods to address these drawbacks.

In various fields, feature selection algorithms play a vital role in providing insights into complex systems. In finance and banking, they can identify relevant variables for credit scoring, fraud detection, and risk management. In environmental and energy issues, they can help to predict environmental pollution levels and energy consumption patterns for efficient resource management. For strategic decision support, feature selection can identify key performance indicators and critical success factors, guiding decision-makers in identifying effective strategies. Finally, in production and scheduling topics, feature selection can optimize production processes, minimize waste, and improve scheduling efficiency.

The proposed feature selection technique possesses a versatile applicability across various scientific and technological domains. The method developed herein finds relevance and utility in numerous scientific and technological fields where data mining plays a pivotal role. Particularly when confronted with an increase in the number of features, the feature selection algorithm proposed holds promise as a means to mitigate computational costs. However, the drawback of this approach lies in its reliance on a single-layer neural network, as introducing additional layers leads to increased network complexity and extended processing times. Consequently, for future research, it is advisable to explore alternative learning methods such as SVM, XGBoost, and Naive Bayes instead of a single-layer neural network. A comparative assessment of these alternative techniques can provide valuable insights into the performance of FSOCP.

REFERENCES

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., . . . Zheng, X. (2016). TensorFlow: a system for large-scale machine learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (s. 265-283). Savannah, GA, USA: USENIX Association.
- Aghasi, A., Abdi, A., Nguyen, N., & Romberg, J. (2017). Net-Trim: Convex Pruning of Deep Neural Networks with Performance Guarantee. *Advances in Neural Informaation Processing Systems*, 3180-3189.
- Agrawal, A., Verschueren, R., Diamond, S., & Boyd, S. (2018). A Rewriting System for Convex Optimization Problems. *Journal of Control and Decision*, 42-60.
- Akadi, A., Ouardighi, A., & Aboutajdine, D. (2008). A Powerful Feature Selection Approach Based on Mutual Information. *International Journal of Computer Science and Network Security*, 8(4), s. 116.
- Ali, M. A., Ucuncu, D., Karadayi Atas, P., & Ozogur - Akyuz, S. (2020). Classification of Motor Imagery Task by Using Novel Ensemble Pruning Approach. *IEEE Transactions on Fuzzy Systems*, 85-91.
- Alizadeh, F., & Goldfarb, D. (2003). Second-order cone programming. *Math. Program*, 3-51.
- Alvear-Sandoval, R. F., & Figueiras-Vidal, A. R. (2015). Does Diversity Improve Deep Learning? *23rd European Signal Processing Conference (EUSIPCO)*. Nice.
- Antonio, B., Moroni, D., & Martinelli, M. (2023, 08 29). Efficient adaptive ensembling for image classification. *Expert Systems*, s. 1-12.
- Anwar, S., Hwang, K., & Sung, W. (2017, May). Structured Pruning of Deep Convolutional Neural Networks. *ACM Journal on Emerging Technologies in computing Systems(JETC)*, 13(3).
- Ardakani, A., Condo, C., & Gross, W. J. (2017). Activation Pruning of Deep Convolutional Neural Networks. *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Montreal: IEEE.
- Azimi, J. F. (2009). "Adaptive Cluster Ensemble Selection". *International Joint Conference on Artificial Intelligence*, (pp. 992-997).
- Baez, J. C., Fritz, T., & Leinster, T. (2011). A Characterization of Entropy in Terms of Information. *Entropy*, 13(11), 1945-1957.
- Battiti, R. (1994). Using Mutual Information for Selecting Features in Supervised

- Neural Netlearning. *IEEE Transactions on Neural Networks*, 5(4), 537-550.
- Beck, A., & Eldar, Y. C. (2013). Sparsity Constrained Nonlinear Optimization: Optimality Conditions and Algorithms. *SIAM Journal on Optimization*, 23(3), s. 1480-1509.
- Beck, A., & Hallak, N. (2018, 04 25). Optimization problems involving group sparsity terms. *Mathematical Programming*, s. 39-67.
- Beck, A., & Hallak, N. (2018, 02 20). Proximal Mapping for Symmetric Penalty and Sparsity. *SIAM Journal on Optimization*, s. 496-527.
- Beck, A., & Vaisbourd, Y. (2018). The Sparse Principal Component Analysis Problem: Optimality Conditions and Algorithms. *Journal of Optimization Theory and Applications*, 1(170), s. 119-143.
- Berikov, V. (2014). Weighted ensemble of algorithms for complex data clustering. *Pattern Recognition Letters*, 38 , 99–106.
- Bertsimas, D., Pauphilet, J., & Parys, B. V. (2021, 11 02). Sparse Classification: a scalable discrete optimization perspective. *Machine Learning*, s. 3177-3209.
- Blalock, D., Ortiz, J., Frankle, J., & Gutttag, J. (2020). What is the state of Neural Network Pruning. *Proceedings of the 3rd MLSys Conference*. Austin, TX, USA.
- Bolon-Canedo, V. S.-M.-B. (2012). An ensemble of filters and classifiers for microarray data classification. *Journals Pattern Recognition*, 45, 531-539.
- Bolón-Canedo, V. S.-M.-B. (2013). A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, 34(3), 483-519.
- Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(2), 123-140.
- Brown, G., Pocock, A., Zhao, M. J., & Lujan, M. (2012). Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *The Journal of Machine Learning Research*, 27-66.
- Burnham, K. P., & Anderson, D. R. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. . Springer.
- Byerly, A., Kalganova, T., & Dear, I. (2021, 11 06). No routing needed between capsules. *Neurocomputing*, s. 545-553.
- Cai, D., Zhang, C., & He, X. (2010). Unsupervised feature selection for multi-cluster data. *16th ACM SIGKDD international conference on Knowledge discovery and data mining*, (s. 333–342).
- Caruana, R. N.-M. (2004). Ensemble Selection from Libraries of Models. *ICML '04*

- Proceedings of the twenty-first international conference on Machine learning* (pp. 18--). Banff, Canada: ACM .
- Charkhabi, M. D. (2014). Cluster Ensembles, Majority Vote, Voter Eligibility and Privileged Voters. *International Journal of Machine Learning and Computin*, 4(3), 275-278.
- Chen, X., Lin, Q., Kim, S., Carbonell, J. G., & Xing, E. P. (2012, 06). Smoothing proximal gradient method for general structured sparse regression. *The Annals of Applied Statistics*, 6(2), s. 719-752.
- Chen, Z.-Y., & Fan, Z.-P. (2014, 09). Parallel multiple kernel learning: a hybrid alternating direction method of multipliers. *Knowledge and Information Systems*, 40(3), s. 673-696.
- Cheng, G., Peddinti, V., Povey, D., Manohar, V., Khudanpur, S., & Yan, Y. (2017). An Exploration of Dropout with LSTMs. *Proc. Interspeech*, (s. 1586-1590).
- Cho, K., Merriënboer, B. v., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (s. 1724-1734). Doha, Qatar: Association for Computational Linguistics.
- Chung, F. (1997). Spectral Graph Theory. *American Mathematical Society*, 92.
- Cortes, C., & Vapnik, V. (1995). Support Vector Networks. *Machine Learning*, s. 273-297.
- Cun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal Brain Damage. *Advances in Neural Information Processing Systems* (s. 598-605). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Diamond, S., & Boyd, S. (2016). CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Journal of Machine Learning Research*, 1-5.
- Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *International Workshop on Multiple Classifier Systems* (s. 1-15). Berlin, Heidelberg: Springer.
- Dong, X., Chen, S., & Pan, S. J. (2017). Learning to Prune Deep Neural Networks via Layer-Wise Optimal Brain Surgeon. *Proceedings of the 31st International Conference on Neural Information Processing Systems* (s. 4860-4874). Long Beach, California, USA: Curran Associates Inc.
- Dosovitskiy, A., Beyer, L., Kolesnikov , A., Weissenborn, D., Zhai, X., Unterthiner,

- T., . . . Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations*. Virtual.
- Dougherty, J., Kohavi, R., & Saham, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the 12th International Conference on Machine Learning* (s. 194-202). Morgan Kaufmann.
- Du, M., Yang, F., Zou, N., & Hu, X. (2020). Fairness in Deep Learning: A Computational Perspective. *IEEE Intelligent Systems*, 25-34.
- Duda , R., Hart , P., & Stork, D. (2012). *Pattern Classification*. John Wiley & Sons.
- El Akadi, A. A. (2011). A two-stage gene selection scheme utilizing MRMR filter and GA wrapper. *Knowledge Information Systems*, 26(3), 487-500.
- Fern, X. Z. (2003). Random projection for high dimensional data clustering: A cluster ensemble approach. *International Conference on Machine Learning*. <http://www.aaai.org/Papers/ICML/2003/ICML03-027.pdf> adresinden alındı
- Fern, X. Z. (2008). Cluster Ensemble Selection. *Statistical Analysis and Data Mining*, 1(3), 128-141.
- Ferrari, D. N. (2015). Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. *Information Sciences* , 301, 181–194.
- Fleuret, F. (2004). Fast Binary Feature Selection with Conditional Mutual Information. *Journal of Machine Learning Research*, s. 1531-1555.
- Foret , P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization. *The International Conference on Learning Representations (ICLR)* . Virtual.
- Frankle, J., & Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding Spase, Trainable Neural Networks. *International Conference on Learning Representations*. New Orleans, Louisiane, USA.
- Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. Elsevier Inc.
- Gale, T., Elsen, E., & Hooker, S. (2019). The State of Sparsity in Deep Neural Networks. February 17, 2021 tarihinde <http://arxiv.org/abs/1902.09574> adresinden alındı
- Garipov, T., Izmailov, P., Podoprikin, D., Vetrov, D., & Wilson, A. G. (2018, 12). Loss surfaces, mode connectivity, and fast ensembling of DNNs. *NIPS'18: Proceedings of the 32nd International Conference on Neural Information*

- Processing Systems*, s. 8803-8812.
- Ghaemi, R. S. (2011). A review: accuracy optimization in clustering ensembles using genetic algorithms. *Artificial Intelligence Review* , 35(4), 287-318.
- Ghosh, J. A. (2011). Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4), 305–315.
- Gini, C. (1913). Variability and Mutability, Contribution to the Study of Statistical Distribution. *Journal of the Royal Statistical Society*, s. 619-622.
- Green, S., Vineyard, C. M., & Koç, Ç. K. (2018). Mathematical Optimizations for Deep Learning. *Cyber-Physical Systems Security* (s. 69-92). içinde Springer Nature Switzerland AG 2018.
- Guan, D. Y.-K. (2014). A Review of Ensemble Learning Based Feature Selection. *IETE Technical Review*, 31(3), 190-198.
- Guan, D., Yuan, W., Lee, Y. K., & Rasel, M. K. (2014). A Review of Ensemble Learning based Feature Selection. *IETE Technical Review*, 31(3), 190-198.
- Guo, B., & Nixon, M. S. (2009). Gait Feature Subset Selection by Mutual Information. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*.
- Guyon, I. E. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3, 1157-1182.
- Guyon, I., Nikravesh, M., Gunn, S., & Zadeh, L. A. (2006). *Feature Extraction: Foundations and Applications*. Springer.
- Guzman-Rivera, A., Batra, D., & Kohli, P. (2012). Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. *Neural Information Processing Systems (NIPS)*.
- Güldoğuş, B. Ç., Abdullah, A. N., Ali, M. A., & Akyüz, S. (2023). Autoselection of the Ensemble of Convolutional Neural Networks with Second-Order Cone Programming. Available at SSRN: <https://ssrn.com/abstract=4382753>.
- Hall, M., & Smith, L. (1999). Feature Selection for Machine Learning: Comparing a Correlation Based Filter Approach to the Wrapper. *FLAIRS Conference*, (s. 235-239).
- Han, S., Mao, H., & Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *4th International Conference on Learning Representations*. San Juan, Puerto Rico: ICLR.

- Han, S., Mao, H., & Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *4th International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.
- Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning Both Weights and Connections for Efficient Neural Network. *Neural Information Processing Systems Conference (NIPS)*, (s. 1135-1143).
- Hanson, S. J., & Pratt, L. (1989). Comparing biases for minimal network construction with back-propagation. *Advances in Neural Information Processing Systems* (s. 177-185). San Francisco, CA, USA: Morgan Kaufmann Publishers.
- Hanson, S. J., & Pratt, L. (1989). Comparing Biases for Minimal Network Construction with Back-Propagation. *Advances in Neural Information Processing Systems*, 177-185.
- Hassibi, B., & Stork, D. G. (1992). Second order derivatives for network pruning: optimal brain surgeon. *5th International Conference on Neural Information Processing Systems (NIPS)* (pp. 164-171). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- He, X., Cai, D., & Niyogi, P. (2005). Laplacian Score for Feature Selection. *Advances in Neural Information Processing Systems*, 507-514.
- He, Y., Zhang, X., & Sun, J. (2017). Channel Pruning for Accelerating Very Deep Neural Networks. *IEEE International Conference on Computer Vision (ICCV)*, (s. 1398-1406). Venice, ITALY.
- Higham, N. J. (1990). Analysis of the Cholesky Decomposition of a Semidefinite Matrix. *Reliable Numerical Computation*, 161-185.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. arXiv preprint arXiv:1207.0580.
- Ho, T. K. (1998). The random subspace method for constructing decision forests., *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8), 832-844.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 1735-1780.
- Hosmer, D. W., & Lemeshow, S. (2000). *Applied Logistic Regression*. John Wiley & Sons.

- Hu, Y., Sun, S., Li, J., Wang, X., & Gu, Q. (2018). A Novel Channel Pruning Method for Deep Neural Network Compression.
<https://dblp.org/rec/bib/journals/corr/abs-1805-11394> (Dü.). içinde *abs/1805.11394*. CoRR.
- Ibadulla, R., Chen, T. M., & Reyes-Aldasoro, C. C. (2023). FatNet: High-Resolution Kernels for Classification Using Fully Convolutional Optical Neural Networks. *AI*, s. 361-374.
- Izmailov, P., Podoprikin, D., Garipov, T., Vetrov, D. P., & Wilson, A. G. (2018). Averaging Weights Leads to Wider Optima and Better Generalization. *Conference on Uncertainty in Artificial Intelligence*. Monterey, California, USA.
- Jain, A. M. (1999). Data Clustering: A Review. *ACM Computing Surveys*, 3, 31.
- Jakulin, A. (2005). Machine Learning Based on Attribute Interactions. *PhD Dissertation*. University of Ljubljana.
- Jia, J. X. (2011). Bagging-based spectral clustering ensemble selection. *Pattern Recognition Letters*, 32, 1456-1467.
- Jia, J. X. (2012). Similarity-based Spectral Clustering Ensemble Selection. *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012)* (pp. 1071-1074). IEEE.
- Jing, L. T. (2015). Stratified feature sampling method for ensemble clustering of high dimensional data. *Pattern Recognition*, in press.
- Jordao, A., Kloss, R., Yamada, F., & Schwartz, W. R. (2018). Pruning Deep Neural Networks using Partial Least Squares. CoRR.
- Kato, H., & Fukushima, M. (2007). An SQP-type Algorithm for Nonlinear Second-Order Cone Programs. *Optimization Letters*, 1(2), 129-144.
- Kim, S., & Kojima, M. (2001). Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15(3), 201-224.
- Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020). Big Transfer (BiT): General Visual Representation Learning. *ECCV 2020: Computer Vision – ECCV 2020*. Springer.
- Kotsiantis, S., & Kanellopoulos, D. (2006). Discretization Techniques: A Recent Survey. *GESTS International Transactions on Computer Science and Engineering*.

- Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images*.
Toronto, CANADA: University of Toronto.
- Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1), 79-86.
- Kuncheva, L. H. (2004). Using diversity in cluster ensembles. *IEEE, International Conference on Systems, Man, and Cybernetics—SMC*, 2, s. 1214–1219.
- Kuncheva, L. S. (2006). Experimental Comparison of Cluster Ensemble Methods. *Information Fusion, 2006 9th International Conference on* (pp. 1-7).
Florence: IEEE.
- Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017, 12). Simple and scalable predictive uncertainty estimation using deep ensembles. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, s. 6405-6416.
- Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *31st International Conference on Neural Information Processing Systems (NIPS'17)* (s. 6405-6416). Long Beach, California, USA: Curran Associates Inc. .
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 436-444.
- LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal Brain Damage. *NIPS*, (s. 598-605).
- Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Sohl-Dickstein, J., & Pennington, J. (2019). Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. *33rd Conference on Neural Information Processing Systems (NeurIPS)*. Vancouver, CANADA.
- Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., & Batra, D. (2015). *Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks*.
arXiv:1511.06314.
- Lee, S., Purushwalkam, S., Cogswell, M., Ranjan, V., Crandall, D., & Batra, D. (2016). Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles. *NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems*, (s. 2127-2135). Barcelona.
- Lewis, D. (1992). Feature Selection and Feature Extraction for Text Categorization. *Proceedings of the Workshop on Speech and Natural Language*.
- Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., & Tang, J. (2018). Feature

- Selection: A Data Perspective. *ACM Computing Surveys*, s. 1-45.
- Li, J., Monroe, W., & Jurafsky, D. (2016). Understanding Neural Networks through Representation Erasure. <http://arxiv.org/abs/1612.08220> adresinden alındı
- Li, Y., Hu, S.-J., Yang, W.-J., Sun, G.-Z., Yao, F.-W., & Yang, G. (2009). Similarity-Based Feature Selection for Learning from Examples with Continuous Values. *Advances in Knowledge Discovery and Data Mining. PAKDD 2009* (s. 957–964). Springer, Berlin, Heidelberg.
- Li, Z., Yang, Y., Liu, J., Zhou, X., & Lu, H. (2021). Unsupervised Feature Selection Using Nonnegative Spectral Analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26, s. 1026-1032. Ontario, Canada.
- Lichman, M. (2013). *UCI Machine Learning Repository*. (I. S. University of California, Producer) Retrieved from <http://archive.ics.uci.edu/ml/>
- Lin, D., & Tang, X. (2006). Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion. *Computer Vision - ECCV*.
- Lin, J., Rao, Y., Lu, J., & Zhou, J. (2017). Runtime Neural Pruning. *NIPS*.
- Liu, H. M. (1998). *Feature Selection for Knowledge Discovery and Data Mining*. London: Kluwer Academic Publishers.
- Liu, H., Xin, B., Mu, S., & Zhu, Z. (2019). Pruning the deep neural network by similar function. *IOP Conf. Series: Journal of Physics: Conf. Series*. 1187. IOP Publishing.
- Lobo, M. S., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of Second-Order Cone Programming. *Linear Algebra and Its Applications*, 284, 193-228.
- Mackay, D. J. (2002). *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press.
- Margineantu, D. D. (1997). Pruning Adaptive Boosting. *ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning* (s. 211-218). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. .
- Mariette, J., & Nathalie, V.-V. (2018, 03 06). Unsupervised multiple kernel learning for heterogeneous data integration. *Bioinformatics*, s. 1009-1015.
- Marill, T., & Green, T. (1963). On the Effectiveness of the Receptors in Recognition Systems. *IEEE Transactions on Computers*, s. 11-17.
- Meyer, P. E., & Bontempi, G. (2006). On the Use of Variable Complementarity for Feature Selection in Cancer Classification. *EvoWorkshops 2006: Applications*

- of Evolutionary Computing* (s. 91-102). Springer.
- Meyer, P., Schretter, C., & Bontempi, G. (2008). Information Theoretic Feature Selection in Microarray Data using Variable Complementarity. *IEEE Journal of Selected Topics in Signal Processing*, 2(3), s. 261-274.
- Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning. <https://dblp.org/rec/bib/journals/corr/MolchanovTKAK16>.
- Naldi, M. C. (2013). Cluster ensemble selection based on relative validity indexes. *Data Min Knowl Disc*, 27, 259–289.
- Narendra, P. F. (1977). A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on Computer*, 26(9), 917-922.
- Nesterov, Y., & Nemirovski, A. (1994). Interior Point Polynomial Methods in Convex Programming: Theory and Applications. *Society for Industrial and Applied Mathematics (SIAM)*.
- Nguyen-Tuong, D., Seeger, M., & Peters, J. (2009). Model Learning with Local Gaussian Process Regression. *Advanced Robotics*, 23(15), s. 2015-2034.
- Nie , F., Huang, H., Cai, X., & Ding, C. (2010). Efficient and robust feature selection via joint $\ell_{2,1}$ -norms minimization. *NIPS - Proceedings of the 23rd International Conference on Neural Information Processing Systems* (s. 1813–1821). Vancouver, British Columbia, Canada: Curran Associates Inc.
- Nishiyama, T. (2020). Convex Optimization on Functionals of Probability Densities. Center for Open Science. doi:10.31219/osf.io/8nzum
- Oguntimilehin, A., Olusola, A., & Innocent, O. (2019). Towards Achieving Optimal Performance using Stacked Generalization Algorithm: A Case Study of Clinical Diagnosis of Malaria Fever. *Int. Arab J. Inf. Technology*, 16, 1047-1081.
- Özöğür-Akyüz, S., & Otar, B. Ç. (2017). Ensemble Clustering Selection by Optimization of Accuracy-Diversity Trade Off. *25th Signal Processing and Communications Applications (SIU)* (s. 1-4). Antalya, Turkey: IEEE.
- Özöğür-Akyüz, S., Otar, B. Ç., & Karadayı Atas, P. (2020). Ensemble Cluster Pruning via Convex Concave Programming. *Computational Intelligence*, 36(1), 297-319.
- Pearl, J. (1983). *Heuristics*. Addison-Wesley.
- Peng, H., Long , F., & Dinc, C. (2005). Feature Selection Based on Mutual

- Information Criteria of Max-Dependency, Max-Relevance and Min-Redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, s. 1226-1238.
- Polikar, R. (2012). Ensemble Learning. *Ensemble Machine Learning* (s. 1-34). içinde Boston, MA, USA: Springer US.
- Quach, K. G., Duong, C. N., Luu, K., & Bui, T. D. (2017). Non-convex online robust PCA: Enhance sparsity via ℓ_p -norm minimization. *Computer Vision and Image Understanding*, s. 126-140.
- Ramirez, C., Kreinovich, V., & Argaez, M. (2013). Why l_1 is a Good Approximation to l_0 : A Geometric Explanation. *Journal of Uncertain Systems*, 7.
- Robnik-Sikonja, M., & Kononenko, I. (2003). Theoretical and Empirical Analysis of ReliefF and RReliefF. *Machine Learning*, 53, 23-69.
- Saeys, Y. A. (2008). Robust feature selection using ensemble feature selection techniques,. *Proc. of the European conference on machine learning and knowledge discovery in databases*, 2, pp. 313-325.
- Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8:e1249.
- Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*(8).
- Sarumathi, S. S. (2013). A Review: Comparative Analysis of Different Categorical Data Clustering Ensemble Methods. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 7(12), 980-990.
- Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27, 623-656.
- Shannon, C. E. (2001). A Mathematical Theory of Communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3-55.
- Sheng, Y., Li, C., Wang, J., Deng, H., & Zhao, Z. (2015). A Novel Pruning Model of Deep Learning for Large-Scale Distributed Data Processing . *Proceedings of APSIPA Annual Summit and Conference*.
- Sigletos, G., Paliouras, G., Spyropoulos, C. D., & Hatzopoulos, M. (2005). Combining Information Extraction Systems Using Voting and Stacked Generalization. *Journal of Machine Learning*, 6, 1751-1782.
- Singh, V. M. (2010). Ensemble clustering using semidefinite programming with

- applications. *Machine Learning*, 79, 177–200.
- Srinivas, S., & Babu, R. V. (2015). Data-free parameter pruning for Deep Neural Networks. CoRR abs/1507.06149.
- Strehl A., G. J. (2002). Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3, 583-617.
- Ström, N. (1997). Phoneme Probability Estimation with Dynamic Sparsely Connected Artificial Neural Networks. *The Free Speech Journal*(5), 1-41.
- Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017). Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *IEEE International Conference on Computer Vision (ICCV)* (s. 843-852). Venice, Italy: IEEE.
- Suzuki, T., & Tomioka, R. (2011, 06 03). SpicyMKL: a fast algorithm for Multiple Kernel Learning with thousands of kernels. *Machine Learning*, s. 77-108.
- Tamon, C., & Xiang, J. (2000). On the boosting pruning problem. *European Conference on Machine Learning* (s. 404-412). Springer.
- Tamon, C., & Xiang, J. (2000). On the Boosting Pruning Problem. *ECML '00: Proceedings of the 11th European Conference on Machine Learning* (s. 404-412). Barcelona, Spain: Springer.
- Tang, S., & Han, J. (2015). A Pruning Based Method to Learn Both Weights and Connections for LSTM.
<https://nlp.stanford.edu/courses/cs224n/2015/reports/2.pdf>.
- Thakur, S. (2018). Optimization of Convolutional Neural Networks via Distinctiveness Pruning. *ABCs*.
- Tibshirani, R. (1996). Regression Shrinkage and Selection via The Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, s. 267-288.
- Topchy, A. M.-B. (2004). Adaptive clustering ensembles. *Pattern Recognition*, 2004. *ICPR 2004. Proceedings of the 17th International Conference on*. 1, pp. 272 - 275. IEEE.
- Tuv, E. B. (2009). Feature selection with ensembles, artificial variables, and redundancy elimination. *Journal of Machine Learning Research*, 10, 1341-1366.
- Vainer, I. K. (2011). Obtaining scalable and accurate classification in large scale spatio temporal domains. *Knowledge Information Systems*, 29(3), 527-564 .
- Vega-Pons, S. . (2015). On pruning the search space for clustering ensemble

- problems. *Neurocomputing*, 150, 481–489.
- Vega-Pons, S. C.-M.-S. (2008). Weighted Cluster Ensemble Using a Kernel Consensus Function. In *Progress in Pattern Recognition, Image Analysis and Applications, Lecture Notes in Computer Science* (Vol. 5197, pp. 195-202). doi:10.1007/978-3-540-85920-8_24
- Vega-Pons, S. C.-M.-S. (2010). Weighted partition consensus via kernels. *Pattern Recognition*, 43, 2712–2724.
- Vega-Pons, S. R.-S. (2011). A Survey of Clustering Ensemble Algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25,(3), 337-372.
- Vermesan, O., & Friess, P. (2014). *Internet of Things- from Research and Innovation to Market Deployment*. River Publishers Aalborg.
- Vidal-Naquet, & Ullman. (2003). Object recognition with informative features and linear classification. *IEEE International Conference on Computer Vision*, (s. 281-288).
- Wan, R., Xiong, H. ., Li, X., Zhu, Z., & Huan, J. (2019). Towards Making Deep Transfer Learning Never Hurt. *2019 IEEE International Conference on Data Mining (ICDM)* (s. 578-587). IEEE.
- Wang, J., & Chen, Y. (2022, 11 12). Generalization in Transfer Learning. *Introduction to Transfer Learning*, s. 175-202.
- Wang, X., Liu, X., Japkowicz, N., & Matwin, S. (2014). Ensemble of Multiple Kernel SVM Classifiers. *Canadian Conference on Artificial Intelligence* (s. 239-250). Springer.
- Wen, T.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., . . . Young, S. J. (2016). Conditional Generation and Snapshot Learning in Neural Dialogue Systems. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (s. 2153-2162). Austin, TX, USA: Association for Computational Linguistics.
- Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning Structured Sparsity in Deep Neural Networks. *30th Conference on Neural Information Processing Systems (NIPS)*. Barcelona.
- Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning Structured Sparsity in Deep Neural Networks. *Advances in Neural Information Processing Systems*, 2074-2082.

- Whitney, A. W. (1971). A Direct Method of Nonparametric Measurement Selection. *IEEE Transactions on Computers*, s. 1100-1103.
- Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, 5(2), 241-259.
- Wright, S. (1965). The interpretation of population structure by f-statistics with special regard to Systems of Mating. *Evolution*, s. 395-420.
- Xu, H., Caramanis, C., & Mannor, S. (2011). Sparse Algorithms are not stable: A no-free-lunch theorem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1), 187-193.
- Yamashita, R., Nishio, M., Do, R. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 611-629.
- Yang, F. L. (2014). Exploring the diversity in cluster ensemble generation: Random sampling and random projection. *Expert Systems with Applications*, 41, 4844–4866.
- Yang, H., & Moody, J. (1999). Data Visualization and Feature Selection: New Algorithms for Nongaussian Data. *NIPS*, (s. 687-693).
- Yu, L., & Liu, H. (2003). Feature Selection for High Dimensional Data: A Fast Filter Correlation Based Filter Solution. *ICML*, (s. 856-863).
- Yu, Z. L.-S. (2014). Hybrid clustering solution selection strategy. *Pattern Recognition*, 47, 3362–3375.
- Zhang, X., Liu, Z., & Liu, S. (2012). A trust region SQP-filter method for nonlinear second-order cone programming. *Computers&Mathematics with Applications*, 63(12), 1569-1576.
- Zhang, Y. D. (2008). Gene selection algorithm by combining relief and mrmr. *BMC Genomics* 9 (Suppl 2), S-27.
- Zhang, Y., Burer, S., & Street, N. W. (2006). Ensemble Pruning Via Semi-definite Programming. *Journal of Machine Learning Research*(7), s. 1315-1338.
- Zhang, Y., Burer, S., & Street, W. N. (2006). Ensemble Pruning Via Semi-definite Programming. *Journal of Machine Learning Research*, 1315–1338.
- Zhao, Z., & Liu, H. (2007). Spectral Feature Selection for Supervised and Unsupervised Learning. *Proceedings of the 24th International Conference on Machine Learning*.
- Zheng, X., Sun, X., & Li, D. (2011). Nonconvex quadratically constrained quadratic programming: best DC decompositions and their sdp representations. *Journal*

of Global Optimization, s. 695-712.

Zhong C., Y. X. (2015). A clustering ensemble: Two-level-refined co-association matrix with path-based transformation. *Pattern Recognition*, 48, 2699–2709.

Zhou, J., Wang, X., Li, P., & Xu, W. (2016). Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation. *Transactions of the Association for Computational Linguistics*(4), 371-383.

Zhou, Z. (2018). Distinctiveness Reduction Technique using Different Threshold Angles on Convolutional Neural Network. *ABCs*.

Zhu, M., & Gupta, S. (2018). To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. *6th International Conference on Learning Representations*. Vancouver, BC, Canada.

