Solving Bongard Problems with Conceptual Transfer Learning via Multi-Head CNN

Ankit Juneja
School of Computer Science
University of Galway

Supervisor(s)
Dr. James McDermott

In partial fulfillment of the requirements for the degree of
MSc in Computer Science (Data Analytics)
31/08/2023

DECLARATION I, Ankit Juneja, hereby declare that this thesis, titled "Solving Bongard Problems with Conceptual Transfer Learning via Multi-Head CNN", and the work presented in it are entirely my own except where explicitly stated otherwise in the text, and that this work has not been previously submitted, part or whole, to any university or institution for any degree, diploma, or other qualification.

Signature: AJuneja

Abstract

This thesis explores the application of Deep Learning techniques to solve Bongard problems, focusing on the utilization of Transfer learning. Bongard problems are a class of pattern recognition puzzles that require the identification of distinctive features and the generalization of rules from a set of visual examples. By leveraging the power of Deep Learning, these problems can be solved in an automated and efficient manner.

The main goal of this study is to create an architecture that incorporates multiple parallel heads, each concept is associated with Bongard problems. Through an innovative multi-head training process, the model learns proficiency in capturing various geometric, structural, and relational patterns. This in-depth understanding of different patterns becomes the foundation for the next step, where we adapt this knowledge to solve more complex problems such as Bongard problems.

The preparation, representation, and encoding of the Bongard problem dataset are all topics covered in this thesis. The model is fully trained to understand the complex concepts and unique traits among different ideas in the dataset.

Keywords: Bongard Problems, Deep Learning,
Supervised Learning, Pattern Recognition,
Dataset Preparation, Model Training, Transfer Learning,
Multi-head CNN.

Contents

List of Figures

Chapter 1

Introduction

Bongard problems are a kind of puzzle that people use to train their minds and improve their pattern recognition skills. They were created by a researcher named Mikhail Bongard. These puzzles come in pairs, and each pair has two groups of shapes. The task is to find a rule or pattern that makes one group different from the other [1].

1.1 Understanding Bongard Problems

In this section, we will try to understand Bongard's problems in depth, In Bongard's problems the shapes might seem similar at first, but there's a specific thing that makes them unique. It could be how they're arranged, the number of sides they have, or something else. Solving Bongard's problems is a complex task for humans as well [2].

For example: In the example given below figure 1.1: On the left side, there's Set A, and on the right side, there's Set B. It might seem similar at first. However, the actual difference between them is that the images in Set A have a special quality – they can be folded to create a cube. On the other hand, the pictures in

Set B don't have this ability; they can't be folded in a way that forms a cube.

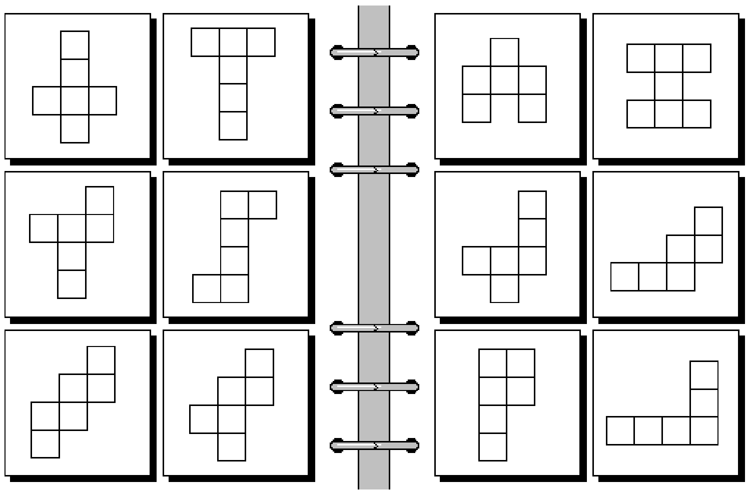


Figure 1.1: Bongard problem
[3]

Another example: In the below example, On the left side images, A cross is placed where the ellipse's longer axis continues However, On the right side images circle is placed where the ellipse's longer axis continues.

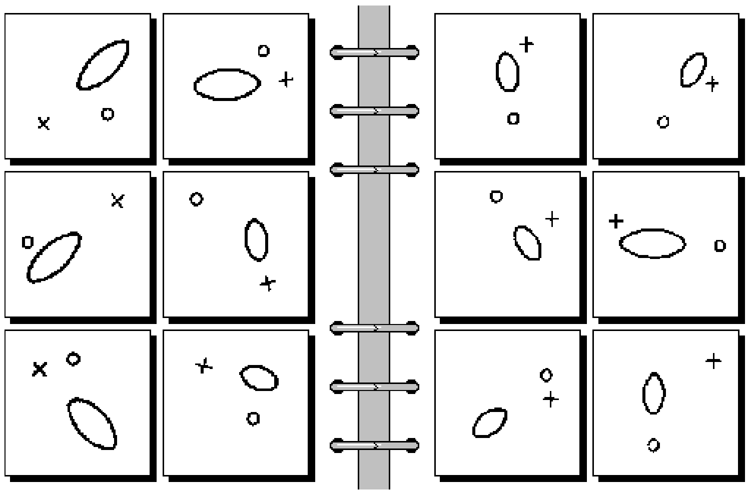


Figure 1.2: Bongard problem
[3]

To solve these problems, we need to carefully look at the shapes and think logically. By comparing the groups of shapes, we can start noticing the special things that separate them. Bongard problems help us become better at paying attention to details and understanding patterns that might not be obvious at the beginning. As we can practice solving more Bongard problems, our ability to think critically and recognize hidden rules.

In a way, Bongard's problems show us how our minds work when we see things and understand patterns. They're like puzzles that teach us about human perception and thinking. Whether we're figuring out the shapes, their positions, or something else, Bongard's problems challenge us to think in new and deeper ways, improving how we see and understand the visual us.

1.2 Motivation and Relevance

I selected the Bongard problem as my research focus because of its complexity. My intention was to check the capabilities of deep learning and how it can solve these complex problems. Bongard problems are known for their challenging nature, and I saw an opportunity to explore how advanced techniques like deep learning could contribute to their solution. This effort matches my curiosity about what technology can do, and it's also a way to explore how deep learning can be used to solve tough problems that need a lot of thinking.

1.3 Research Questions

RQ1: How well my model is able to learn Bongard Problems or concepts?

RQ2) Does my model learn to tell the difference between Concepts and Perform well on Bongard Problems?

RQ3) How does the performance of the cold-start model compare to warm start when both train on the same dataset?

RQ4) How is the performance of the model on multiple concepts, versus individual concepts?

RQ5) Does this type of pre-training help us to achieve good performance on any BPs?

Chapter 2

Background

In this chapter, we'll learn how Artificial Intelligence can solve complex problems. We'll explore the exciting world of Artificial Intelligence (AI) and how it works to analyze patterns, make decisions, and creatively challenges. By understanding these concepts, we'll start learning the basics to understand Bongard's problems and how AI helps us solve those problems. We will also explore how AI is really smart and can learn and think in amazing ways.

2.1 Aritifical Intelligence

Artificial Intelligence (AI) is a field in computer science where we aim to make computers perform tasks exactly in the same way as we humans do. For example: when we look at pictures, our brain recognizes what's in them - AI is also about making computers do something similar.[4] We give them lots of data and help them learn patterns from it. For example, we can teach a computer to tell the difference between pictures of cats and dogs by showing it many examples of both.

AI also makes decisions based on what they've learned. for example, deciding whether to take an umbrella depends on the weather. In this case, we train our

AI with data regarding weather and umbrellas, In what scenarios we are going to use an umbrella or not [4], Based on this data AI learns the pattern of when we should use an umbrella or not. Thereafter, It makes a decision whether we need to use an umbrella or not in this weather.

AI aims to enable computers to do tasks that need intelligence, like recognizing spoken words, understanding languages, and even driving cars. It's like teaching them to think and make smart decisions on their own.[4]. This way, we're making computers smarter by using AI and more capable of doing tasks that we used to think only humans could do. Moreover, the goal is to make computers do things that require human-like intelligence, and in the process, we're making our lives easier and more interesting.

2.2 How AI exactly solves the problems

Artificial Intelligence (AI) solves problems exactly in the same way as our human brains do, but with special techniques for computers to process a lot of information faster than us. Here are the steps:

Data Gathering: AI, like us learning from experiences, learns from data. It collects a lot of information, such as pictures, words, or numbers, to learn the world and detect patterns. Moreover, data is the main part of AI models. The quality and quantity of data play a crucial role in determining how well an AI model performs on a given task.

Feature Extraction: AI breaks data into smaller parts known as "features." For example, with animal pictures, it might notice things like eyes, ears, and tails. These features are like clues that help AI understand and sort things properly [5]. Also, These features are the characteristics of the data that the AI model will use to make predictions or decisions.

Learning: AI uses complex steps, called algorithms, to work with features and find patterns. It's exactly work same as we humans work how to identify things and detect patterns. AI changes its inside processes to understand patterns in data. [5]

Problem Representation: Before solving a problem, AI needs to understand it. It transforms the problem into a form it can understand – for example, preprocess into numbers or symbols. [6]

Algorithmic Processing: AI uses special rules and algorithms (like step-by-step instructions) to manage the problem. These algorithms find solutions based on patterns from data. [6]

Prediction and Decision-Making: AI analyzes the problem and makes predictions or decisions based on what it learned. It can also guess future events, classify things or suggest actions. [7]

Feedback Loop: AI's choices are usually compared to the right answers, so it learns from its errors. This loop helps AI improve over time. In technical terms, it is also known as training loss.

Overall, AI solves problems by handling lots of data, spotting patterns, learning from examples, and using algorithms to make decisions. It's a mix of understanding, learning, and clever handling that lets AI deal with various complex tasks and issues.

2.3 Complexity of AI

In this section we will discuss the complexity of AI, Artificial Intelligence (AI) seems really helpful, but it faces some problems that affect how it grows and works. These things show how hard it is to make an AI model that behaves exactly the same as we humans do.

2.3.1 Challenge when AI has less data to train

When Artificial Intelligence (AI) doesn't have much information or data to learn from, it faces a big problem that affects how well it works. AI needs data to learn and make good choices. Just like we learn things by seeing something and experiencing them, AI learns patterns and behaviors from the information it gets. This way, it can recognize stuff, guess what will happen, and decide wisely.

But when AI doesn't have enough data, it can't fully understand things and its decisions might not be as good. AI needs many examples to learn broadly and handle different situations. If it doesn't have enough data, its guesses and choices might not be very sure or right.

To solve this problem, some people who study AI look for ways to help it do better with less data. They make unique approaches. for example, we need to detect cat vs dog but we don't have many images to train our AI model, so in that case we will download other images which has the same properties like the eyes of cats and dogs, hair of cats and dogs, nails of cats and dogs then we train our AI model with these images, and later on we when we train our model with these images we will pass real images of cats and dogs to differentiate between them. In technical terms, it is also known as Transfer learning. [8]

Overall, having not enough data for training can slow down AI's ability to work well. When there's not much data, its guesses and predictions might not be always right.

2.4 AI and Bongard Problem

In this section, we will see how AI can perform on Bongard problems.

2.4.1 Bongard Problem Solving with AI

As we know the complexity of the Bongrad problem from previous chapters is that it follows a different pattern which sometimes it's very difficult for humans to understand. But in the case of AI, It might perform better because AI can learn multiple patterns from the data that it has been trained and AI has a fast processing speed as compared to humans to recognize patterns if it's trained properly. Moreover, some Bongard problems may involve repetitive tasks of identifying patterns in multiple scenarios, so in this case, AI consistently learns patterns from those scenarios and makes it good prediction for such repetitive tasks. Furthermore, AI can learn things from its mistakes as well by comparing its decisions with correct answers, because Bongard problems usually have correct outputs while training so it that may allow AI to improve by learning from its errors.

2.4.2 Challenges in Solving Bongard Problems with AI:

In the above part, we discuss how can AI perform better in terms of solving Bongard problems, however, there are some reasons why I believe might AI not perform better in solving Bongard problems. Which I will discuss below.

Limited Data for Bongard Problems: To train an AI model so that it can perform better on Bongard problems we should have enough data to train our model, and this data should have all possible variations of which Bongard problem belongs. Unfortunately, we don't have enough data to train our model, In Bongard problem we only have 10 images out of 12 to train the model because 2 images will be used for testing purposes and it's difficult to generate data that is similar to Bongard problems that can cover all variations.

Complex Pattern Recognition: Some Bongard problems are extremely difficult it requires human imagination to solve those problems.

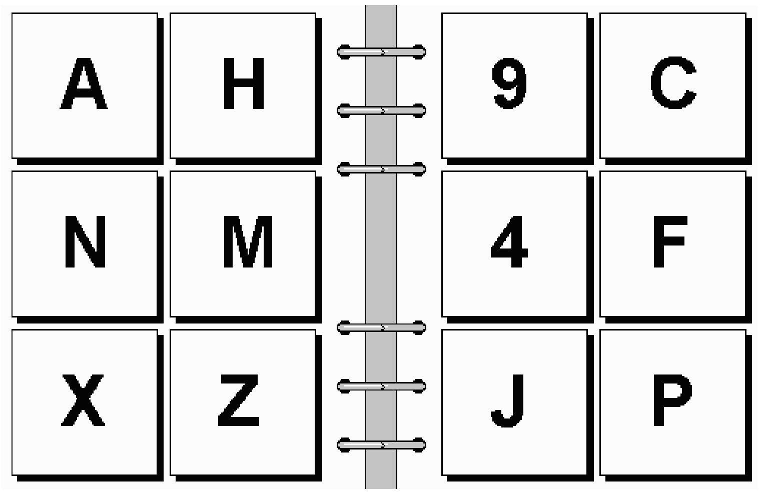


Figure 2.1: Bongard problem
[3]

for example: In figure 2.1 left side images are stable, If we put those alphabets on the floor they will stand (stable), however, images on the right side are unstable if we put those numbers or alphabets in the flat surface it will fall down. So, training these concepts from alphabets or numbers is very difficult for AI.

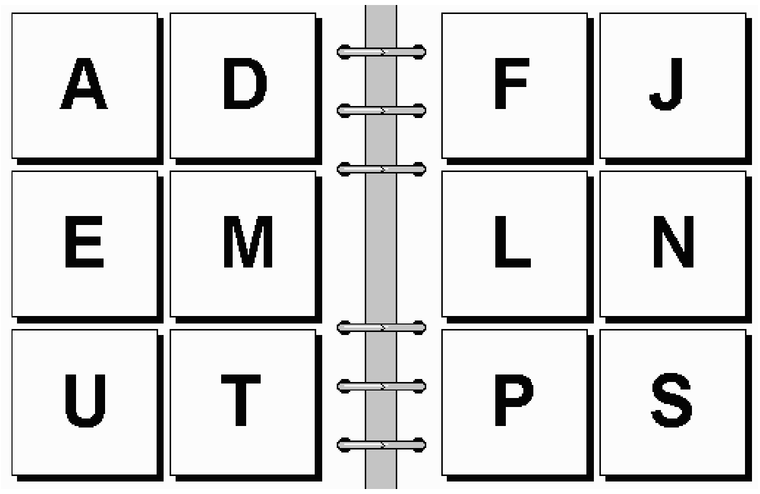


Figure 2.2: Bongard problem
[3]

Similarly, according to figure 2.2, it again has alphabets to differentiate the images into left side vs right side, but this time images on the left side are symmetry and images on are right side are non-symmetric. As a human, we can imagine and differentiate between alphabets whether it's symmetry, no symmetry, or stable or unstable but to train these concepts to AI with the same data is a challenging task.

Chapter 3

Related Work

In this Chapter, we will look at what other researchers have discovered about using AI for solving Bongard problems. We'll start by understanding a concept called "transfer learning," which involves using knowledge from one area to help solve problems in another area. We'll explore a student's work where he used a smart approach to solve Bongard problems by first training it on a dataset for recognizing numbers.

Then, we'll discuss my approach, which is different. Instead of using someone else's approach, I came up with my own ideas (under the guidance of my professor). I created a new dataset specifically for Bongard problems and used it in a unique way. We'll talk about the advantages and challenges of this approach compared to the student's approach.

Throughout this section, we'll see how different approaches fit together and learn about the contributions of each to solving Bongard problems using AI.

3.1 Transfer Learning for Bongard Problem Solving:

In this section, we will explore the concept of transfer learning and its role in solving Bongard's problems using AI. Transfer learning is an approach where we learn from one task and use that learning to a new task. For example: In our childhood, we learned basic mathematics to count fruits and vegetables where we learned how to count things, and now we applied it to count money and other things because we learned the concept of counting. In the case of solving Bongard problems, transfer learning involves adapting a technique that's good at something else, like recognizing numbers, etc. to solve these unique challenges.

3.1.1 Introduction to Transfer Learning:

For a better understanding of transfer learning, Niklas Donges[9] states that "Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another. For example, in training a classifier to predict whether an image contains food, you could use the knowledge it gained during training to recognize drinks." Also, Professor Ryan Ahmed explains this concept in more depth with the help of a real example of how he used transfer learning in his YouTube video [10] where he discussed that he had one pre-trained model which is trained on ImageNet dataset, later on he transferred the trained parameters of Convolution layers in his model and add more dense layers so that he can try on new data such as x-ray images.

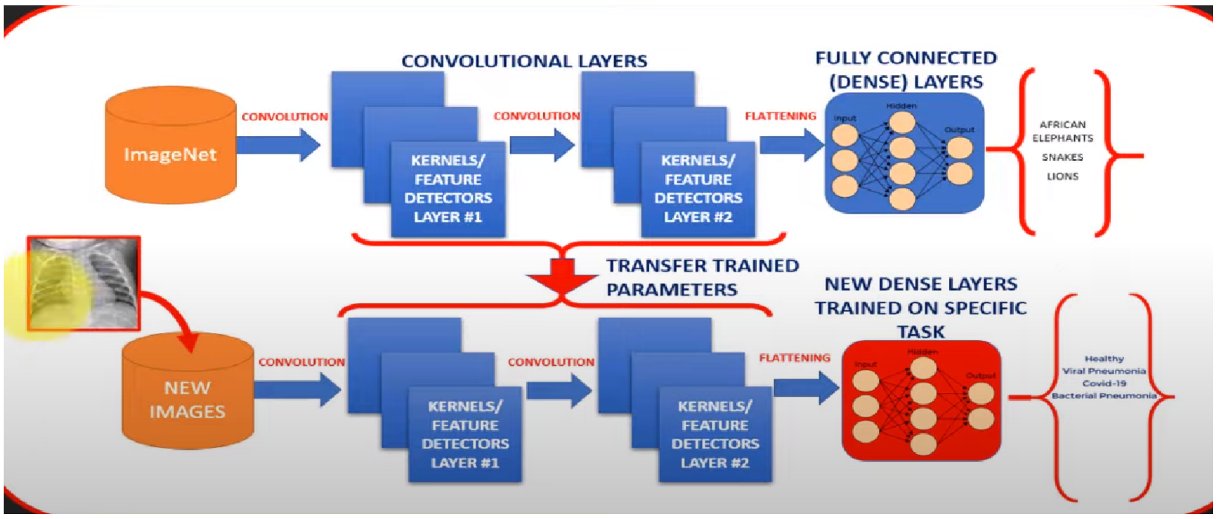

CONVOLUTIONAL LAYERS
FULLY CONNECTED (DENSE) LAYERS
ImageNet
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #1
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #2
FLATTENING
AFRICAN ELEPHANTS
SNAKES
LIONS
TRANSFER TRAINED PARAMETERS
NEW DENSE LAYERS TRAINED ON SPECIFIC TASK
NEW IMAGES
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #1
CONVOLUTION
KERNELS/ FEATURE DETECTORS LAYER #2
FLATTENING
Healthy
Viral Pneumonia
Covid-19
Bacterial Pneumonia

Figure 3.1: Prof Ryan Ahmed Transfer Learning
[10]

3.1.2 Applying Transfer Learning to Bongard Problems:

From the above section, I got an idea of how transfer learning actually works, Later on, I read the article where Sergii Kharagorgiev applied transfer learning to solve Bongard problems [11]. First, he created a synthesis dataset of geometric shapes which is common in Bongard problems, then he trained his model with the same dataset. After that, he applied transfer learning to solve bongard problems. from this article, I got the idea that transfer learning can actually help me with Bongard's problems.

Furthermore, I read a university thesis paper which is from the University of Galway, where one student Mark Fahy wrote a thesis on "Bongard Problems and Modern Neural Networks" [12] where he discussed how he used transfer learning for his thesis and how he solved the bongard problems with the help of transfer learning.

These projects show that transfer learning is a promising approach to deal with the lack of data in Bongard Problems. However, transfer learning can be used in a different way, compared to that of Fahy [12] and Sergii [11] articles, which we will discuss in Section Methodology section.

From the above two articles I was sure that I was going to use the transfer learning approach in my project as well but I had some unique ideas to do with that.

3.1.3 Fine-Tuning the Pre-trained Model:

Sergii used a pre-trained CNN model on a synthetic dataset of geometric shapes. This pre-trained model had learned to extract features from images, like recognizing shapes, edges, and textures. These features capture important information in an image that can be used for classification. Thereafter, he used each of the 12 images in the Bongard problem and passed them through the layers of the pre-trained CNN. This process extracted relevant features from each image [11].

Whereas, Mark Fahy, first trained his model in the MNIST dataset, which is handwritten digits. which helps the model to learn different types of features such as curves, and different numbers which are useful in Bongard problems thereafter when he trained that model he used the trained model to work on bongard problems. [12]

3.1.4 Outcomes and Insights:

In this section, we will see what are the outcomes and results came after their research.

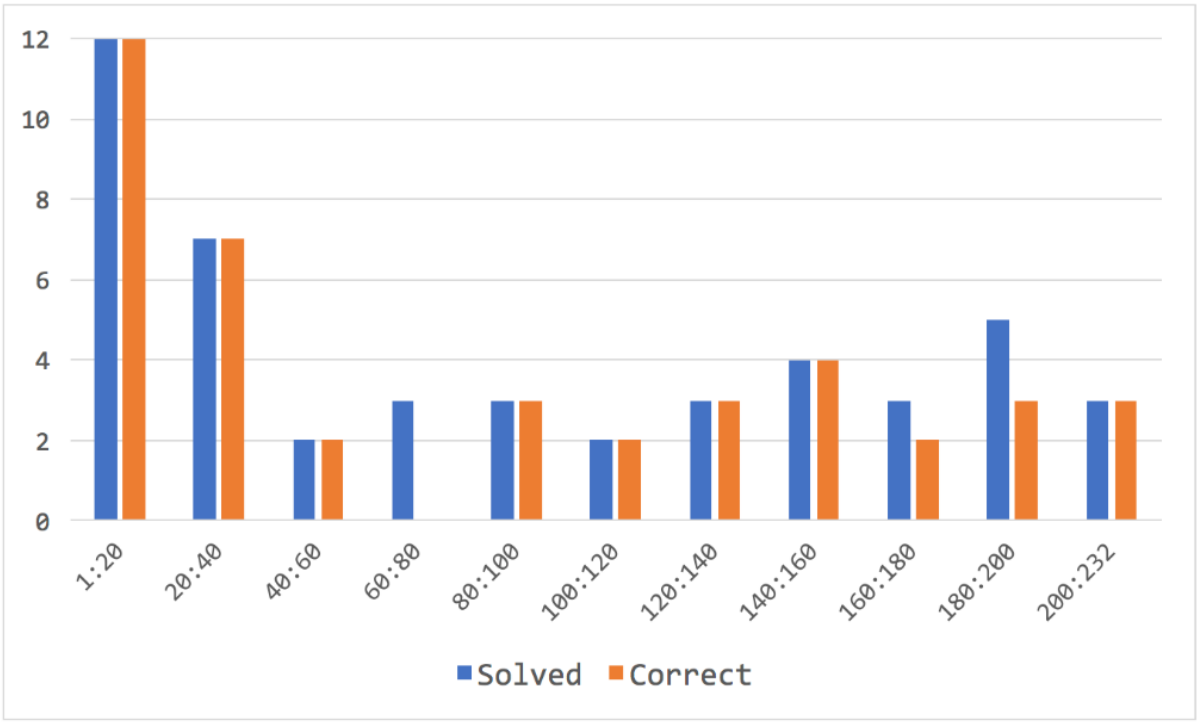
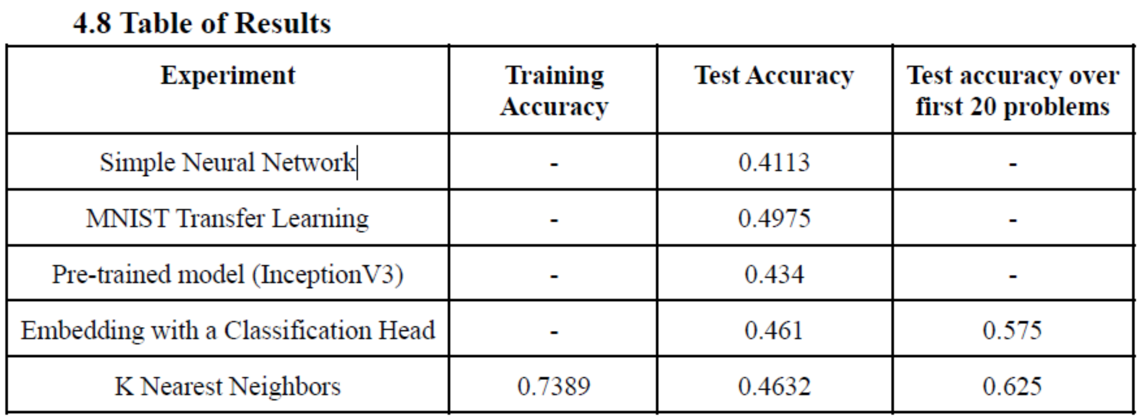


Figure 3.2: Results from Sergii Research
[11]

As Sergii mentioned in his research that he is able to solve only a few Bongard problems by using his approach as shown in figure 3.2. He compare the solved vs correct problem in a batch of 20.



4.8 Table of Results

Experiment	Training Accuracy	Test Accuracy	Test accuracy over first 20 problems
Simple Neural Network	-	0.4113	-
MNIST Transfer Learning	-	0.4975	-
Pre-trained model (InceptionV3)	-	0.434	-
Embedding with a Classification Head	-	0.461	0.575
K Nearest Neighbors	0.7389	0.4632	0.625

Figure 3.3: Results from Mark fahy
[12]

Whereas, according to Mark Fahy's report, it shows that when he used only a simple Neural network he achieved around 41 percent accuracy, and when he used transfer learning through the MNIST dataset he increased by 8 percent accuracy as shown in figure 3.3

3.2 Custom Dataset and Multi-Head CNN Approach:

3.2.1 Concepts creation

From the above section, I got a clear idea that transfer learning is a promising approach for my thesis. So the first step I took was to start looking for the Bongard problems one by one when I looked at all Bongard problems with their solution [3] I realized that there are some Bongard problems that follow the same concepts, for example, In figure 3.4 and figure 3.5 are following the same pattern that is horizontal vs vertical. In figure 3.4 left side images are vertical in direction and right side images are horizontal in direction. However, In the figure, 3.5 circles on the left side are horizontal and the circles on the right side are vertical and vice-versa for rectangles.

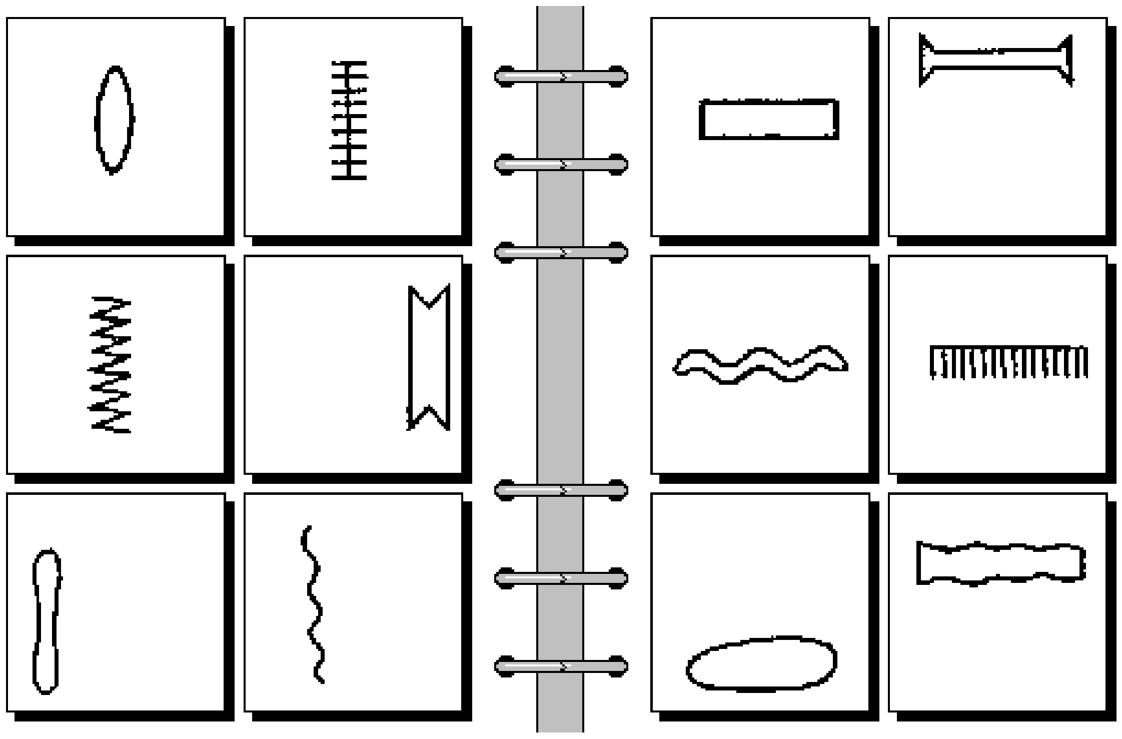


Figure 3.4: Concepts generated for Bongard problems
[3]

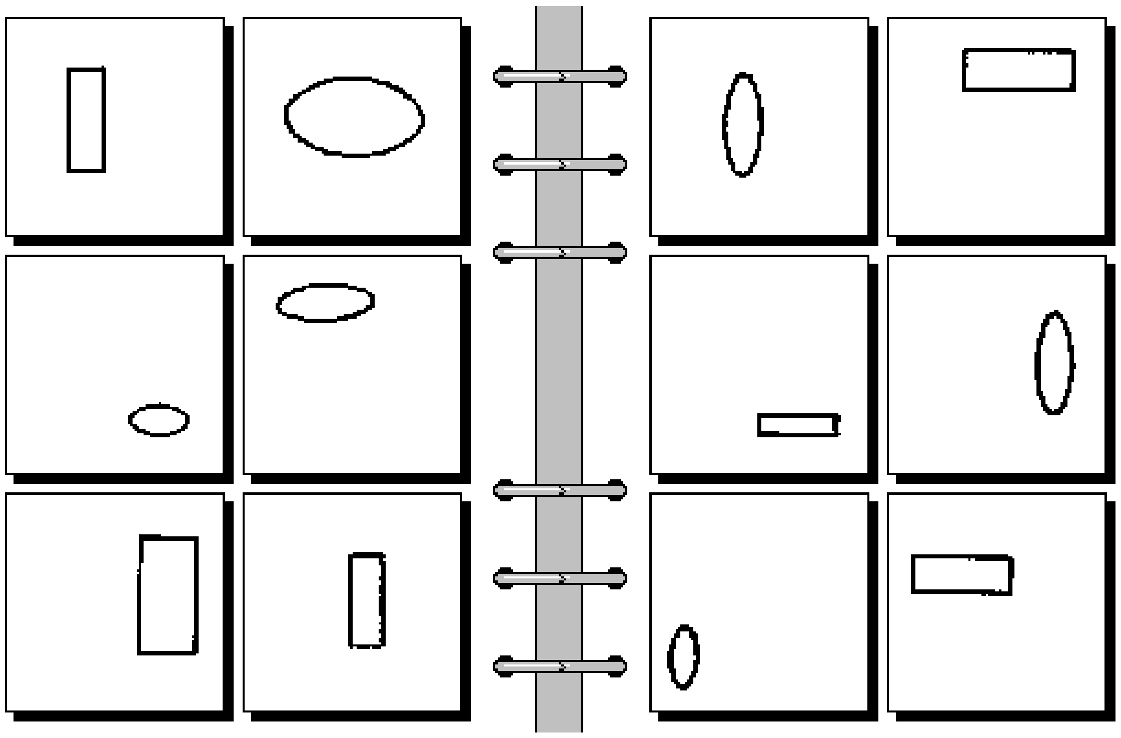


Figure 3.5: Concepts generated for Bongard problems
[3]

Similarly, I looked for all 394 Bongard problems and started making the concepts that belong to Bongard problems, I have made around 70 concepts which I discussed in my data chapter as well. Out of 70 concepts I found 20 concepts that consist of most of the Bongard problems, so my idea is to teach all these concepts to my model if my model is able to classify many things then it can perform very well in real Bongard images, which means I'm planning to use transfer learning in a different way.

3.2.2 Building Multi head CNN

Thereafter, I started building my Multi-head CNN model and I realized I should aim to construct a model with the flexibility to connect multiple layers (heads) from any point. To address this, I explored relevant model architectures that had been made by Functional API in Keras, which allowed me to make non-linear models with arbitrary layer connections [13]. Also, I read one article on Pyimage search [14] website from where I learned the clear difference between multi-label outputs and multi-output models. After all this, I was able to make my multi-head CNN, which I will discuss the details in the methodology section.

3.3 Innovation and Contributions:

In this section, I'll discuss how I learned from previous articles and how my work is different from theirs.

To begin with, I didn't just follow the usual steps I added some new ideas that make solving these problems even more interesting.

First off, I didn't use the same prepared dataset that everyone else was using. I made my own data. I made sure this data had enough variations to see Bongard's problems. So, when our model learned from this data, it got really good at solving Bongard problems.

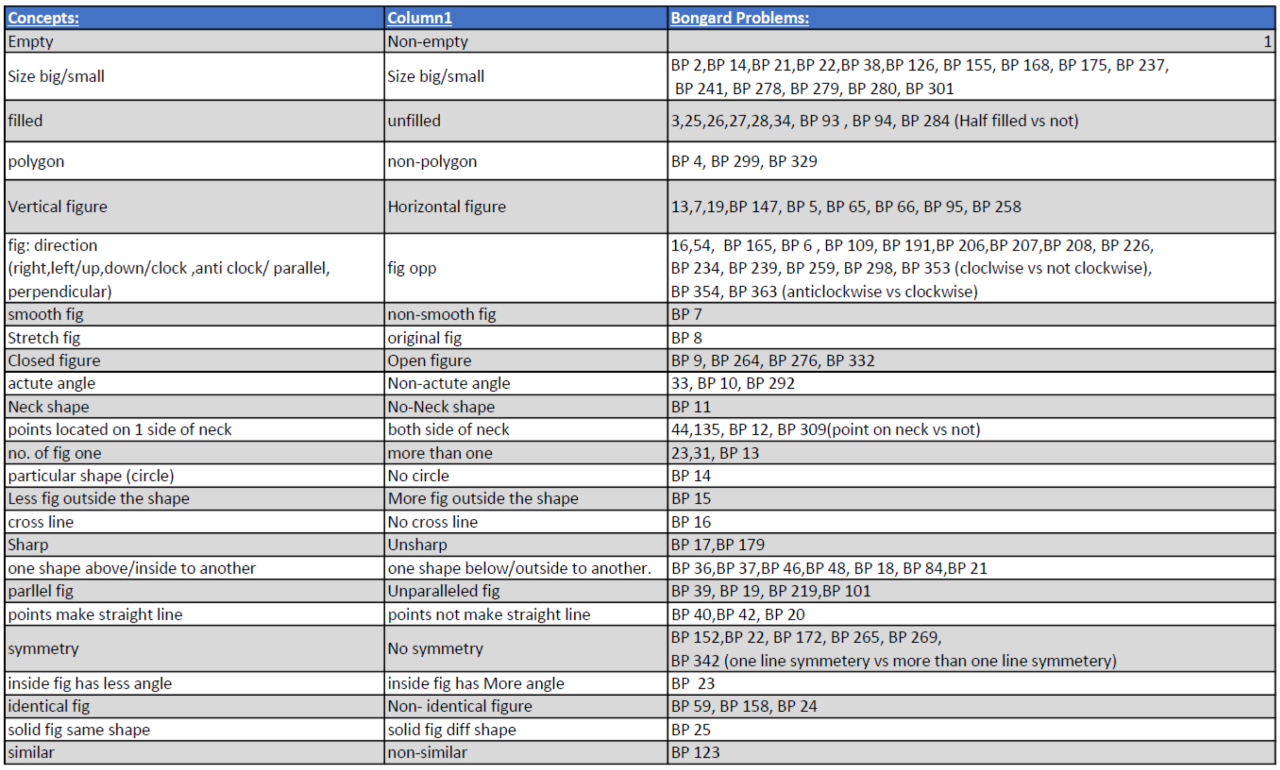
Also, I didn't stop with a regular learning machine approach. I built a clever Multi-head CNN with different heads. Each head learned something important from my dataset.

Moreover, I made our model to learn from my dataset with the other parts. So, when our model looked at Bongard's problems, it wasn't starting from zero. It already had a bunch of useful knowledge.

Chapter 4

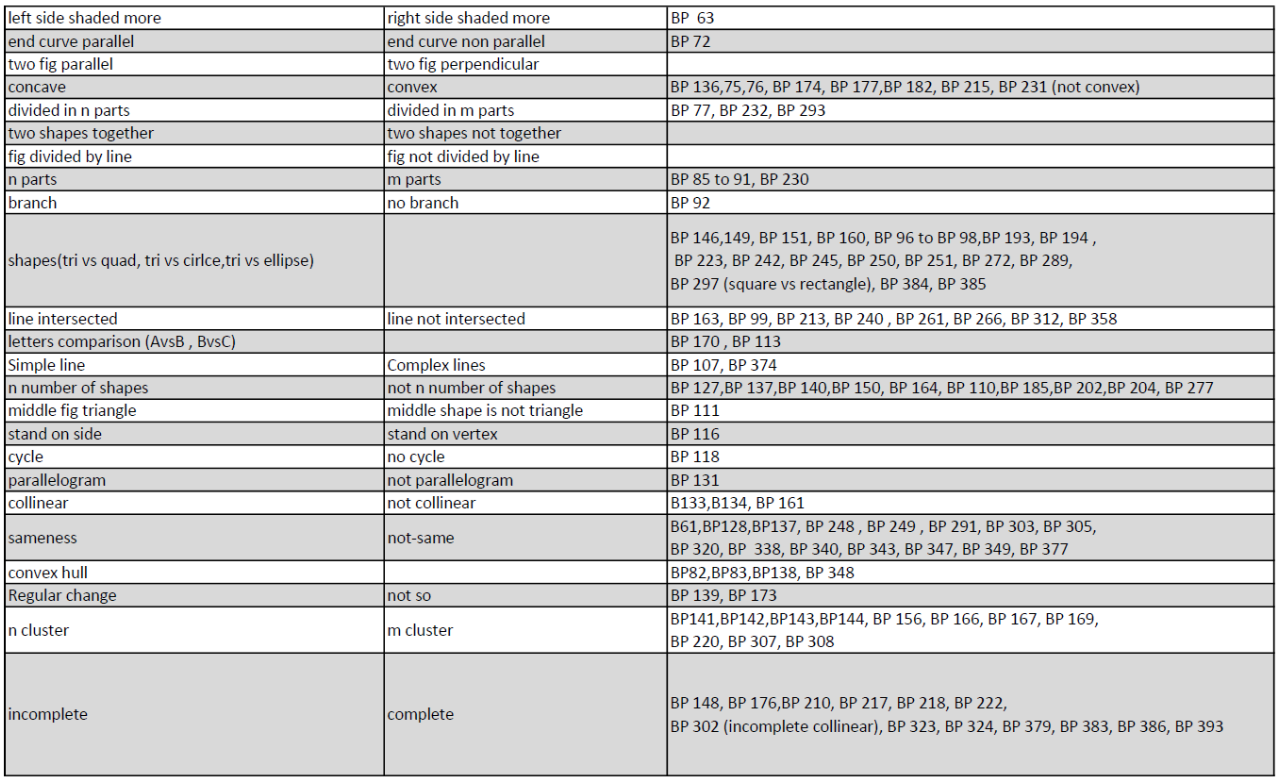
Data

4.1 Concepts generated for Bongard Problems:



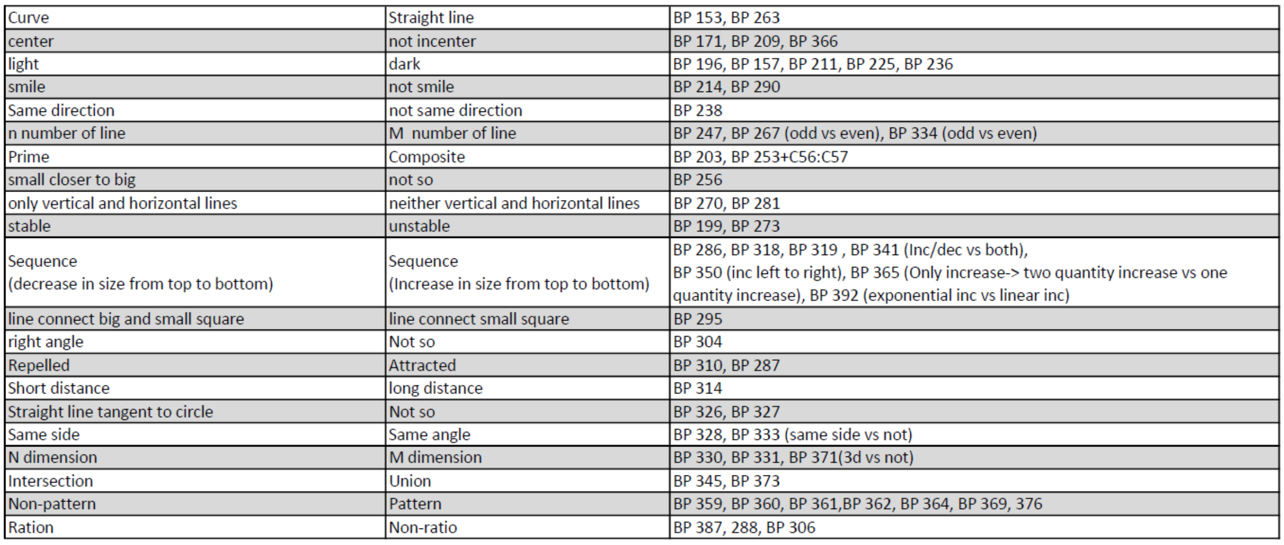
Concepts:	Column1	Bongard Problems:
Empty	Non-empty	1
Size big/small	Size big/small	BP 2,BP 14,BP 21,BP 22,BP 38,BP 126, BP 155, BP 168, BP 175, BP 237, BP 241, BP 278, BP 279, BP 280, BP 301
filled	unfilled	3,25,26,27,28,34, BP 93 , BP 94, BP 284 (Half filled vs not)
polygon	non-polygon	BP 4, BP 299, BP 329
Vertical figure	Horizontal figure	13,7,19,BP 147, BP 5, BP 65, BP 66, BP 95, BP 258
fig: direction (right,left/up,down/clock ,anti clock/ parallel, perpendicular)	fig opp	16,54, BP 165, BP 6 , BP 109, BP 191,BP 206,BP 207,BP 208, BP 226, BP 234, BP 239, BP 259, BP 298, BP 353 (cloclwise vs not clockwise), BP 354, BP 363 (anticlockwise vs clockwise)
smooth fig	non-smooth fig	BP 7
Stretch fig	original fig	BP 8
Closed figure	Open figure	BP 9, BP 264, BP 276, BP 332
actute angle	Non-actute angle	33, BP 10, BP 292
Neck shape	No-Neck shape	BP 11
points located on 1 side of neck	both side of neck	44,135, BP 12, BP 309(point on neck vs not)
no. of fig one	more than one	23,31, BP 13
particular shape (circle)	No circle	BP 14
Less fig outside the shape	More fig outside the shape	BP 15
cross line	No cross line	BP 16
Sharp	Unsharp	BP 17,BP 179
one shape above/inside to another	one shape below/outside to another.	BP 36,BP 37,BP 46,BP 48, BP 18, BP 84,BP 21
parllel fig	Unparalleled fig	BP 39, BP 19, BP 219,BP 101
points make straight line	points not make straight line	BP 40,BP 42, BP 20
symmetry	No symmetry	BP 152,BP 22, BP 172, BP 265, BP 269, BP 342 (one line symmetery vs more than one line symmetery)
inside fig has less angle	inside fig has More angle	BP 23
identical fig	Non- identical figure	BP 59, BP 158, BP 24
solid fig same shape	solid fig diff shape	BP 25
similar	non-similar	BP 123

Figure 4.1: Concepts generated for Bongard problems part 1



left side shaded more	right side shaded more	BP 63
end curve parallel	end curve non parallel	BP 72
two fig parallel	two fig perpendicular	
concave	convex	BP 136,75,76, BP 174, BP 177,BP 182, BP 215, BP 231 (not convex)
divided in n parts	divided in m parts	BP 77, BP 232, BP 293
two shapes together	two shapes not together	
fig divided by line	fig not divided by line	
n parts	m parts	BP 85 to 91, BP 230
branch	no branch	BP 92
shapes(tri vs quad, tri vs cirlce,tri vs ellipse)		BP 146,149, BP 151, BP 160, BP 96 to BP 98,BP 193, BP 194 , BP 223, BP 242, BP 245, BP 250, BP 251, BP 272, BP 289, BP 297 (square vs rectangle), BP 384, BP 385
line intersected	line not intersected	BP 163, BP 99, BP 213, BP 240 , BP 261, BP 266, BP 312, BP 358
letters comparison (AvsB , BvsC)		BP 170 , BP 113
Simple line	Complex lines	BP 107, BP 374
n number of shapes	not n number of shapes	BP 127,BP 137,BP 140,BP 150, BP 164, BP 110,BP 185,BP 202,BP 204, BP 277
middle fig triangle	middle shape is not triangle	BP 111
stand on side	stand on vertex	BP 116
cycle	no cycle	BP 118
parallelogram	not parallelogram	BP 131
collinear	not collinear	B133,B134, BP 161
sameness	not-same	B61,BP128,BP137, BP 248 , BP 249 , BP 291, BP 303, BP 305, BP 320, BP 338, BP 340, BP 343, BP 347, BP 349, BP 377
convex hull		BP82,BP83,BP138, BP 348
Regular change	not so	BP 139, BP 173
n cluster	m cluster	BP141,BP142,BP143,BP144, BP 156, BP 166, BP 167, BP 169, BP 220, BP 307, BP 308
incomplete	complete	BP 148, BP 176,BP 210, BP 217, BP 218, BP 222, BP 302 (incomplete collinear), BP 323, BP 324, BP 379, BP 383, BP 386, BP 393

Figure 4.2: Concepts generated for Bongard problems part 2



Curve	Straight line	BP 153, BP 263
center	not incenter	BP 171, BP 209, BP 366
light	dark	BP 196, BP 157, BP 211, BP 225, BP 236
smile	not smile	BP 214, BP 290
Same direction	not same direction	BP 238
n number of line	M number of line	BP 247, BP 267 (odd vs even), BP 334 (odd vs even)
Prime	Composite	BP 203, BP 253+C56:C57
small closer to big	not so	BP 256
only vertical and horizontal lines	neither vertical and horizontal lines	BP 270, BP 281
stable	unstable	BP 199, BP 273
Sequence (decrease in size from top to bottom)	Sequence (Increase in size from top to bottom)	BP 286, BP 318, BP 319 , BP 341 (Inc/dec vs both), BP 350 (inc left to right), BP 365 (Only increase-> two quantity increase vs one quantity increase), BP 392 (exponential inc vs linear inc)
line connect big and small square	line connect small square	BP 295
right angle	Not so	BP 304
Repelled	Attracted	BP 310, BP 287
Short distance	long distance	BP 314
Straight line tangent to circle	Not so	BP 326, BP 327
Same side	Same angle	BP 328, BP 333 (same side vs not)
N dimension	M dimension	BP 330, BP 331, BP 371(3d vs not)
Intersection	Union	BP 345, BP 373
Non-pattern	Pattern	BP 359, BP 360, BP 361,BP 362, BP 364, BP 369, 376
Ration	Non-ratio	BP 387, 288, BP 306

Figure 4.3: Concepts generated for Bongard problems part 3

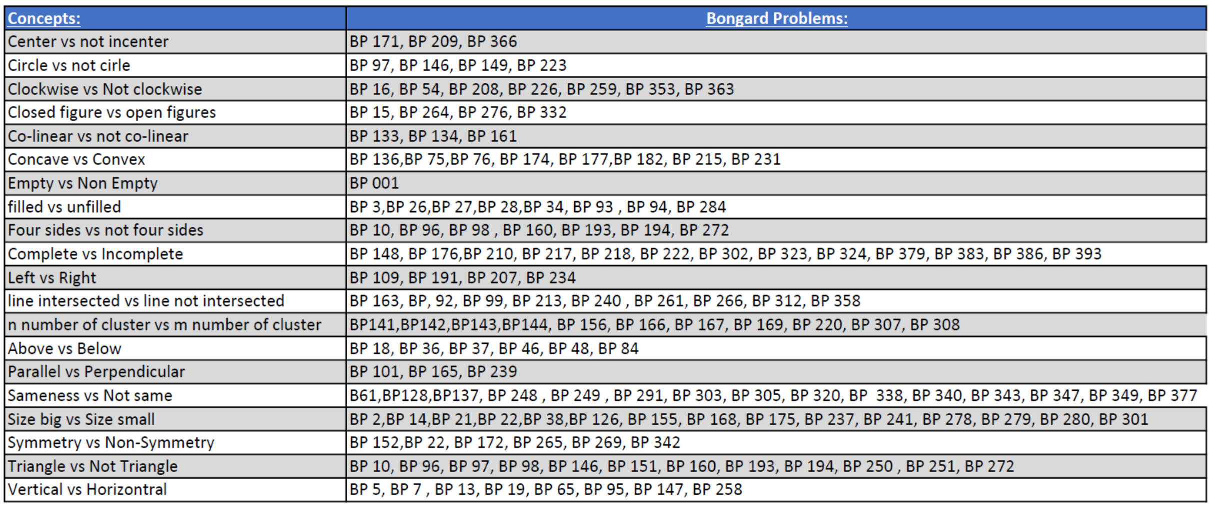
I have identified different concepts that Bongard problems have in common. These concepts focus on things like the size of shapes, their positions, whether they're empty or filled, and other aspects. Each concept group consists of similar Bongard problems, this makes it easier to study how they relate and differ from one concept to another as shown in figure 4.3 Also I have attached my zip file where anyone can download this table. **Click here**

By organizing Bongard's problems using these ideas, I have created a table where I have 70 different concepts to study them more effectively. Also, This can help in identifying patterns and trends that can be used to improve the accuracy of your multi-head CNN model in future as well.

4.2 Concept-Based Dataset Generation:

In this part, I created a dataset for 20 concepts out of the 70. These 20 concepts cover a majority of the Bongard problems, which is around 150. Each concept focuses on different aspects, for example: whether shapes are in the center or not, the shape of circles, clockwise or anticlockwise orientation, closed or open figures, and many more as shown in the figure 4.4. These concepts help me to categorize and understand the Bongard problems.

The selected Bongard problems are connected to these concepts. For example, the concept of "Center vs not incenter" has all those Bongard problems that follow the same concept. Similarly, concepts like the "Symmetry vs Non-Symmetry" group have all those Bongard problems that belong to this concept. These 20 concepts help me to study Bongard's problems in a more organized way. They group together problems that are similar, making it simpler to examine the patterns and differences in these problems.



Concepts:	Bongard Problems:
Center vs not incenter	BP 171, BP 209, BP 366
Circle vs not cirle	BP 97, BP 146, BP 149, BP 223
Clockwise vs Not clockwise	BP 16, BP 54, BP 208, BP 226, BP 259, BP 353, BP 363
Closed figure vs open figures	BP 15, BP 264, BP 276, BP 332
Co-linear vs not co-linear	BP 133, BP 134, BP 161
Concave vs Convex	BP 136,BP 75,BP 76, BP 174, BP 177,BP 182, BP 215, BP 231
Empty vs Non Empty	BP 001
filled vs unfilled	BP 3,BP 26,BP 27,BP 28,BP 34, BP 93 , BP 94, BP 284
Four sides vs not four sides	BP 10, BP 96, BP 98 , BP 160, BP 193, BP 194, BP 272
Complete vs Incomplete	BP 148, BP 176,BP 210, BP 217, BP 218, BP 222, BP 302, BP 323, BP 324, BP 379, BP 383, BP 386, BP 393
Left vs Right	BP 109, BP 191, BP 207, BP 234
line intersected vs line not intersected	BP 163, BP, 92, BP 99, BP 213, BP 240 , BP 261, BP 266, BP 312, BP 358
n number of cluster vs m number of cluster	BP141,BP142,BP143,BP144, BP 156, BP 166, BP 167, BP 169, BP 220, BP 307, BP 308
Above vs Below	BP 18, BP 36, BP 37, BP 46, BP 48, BP 84
Parallel vs Perpendicular	BP 101, BP 165, BP 239
Sameness vs Not same	B61,BP128,BP137, BP 248 , BP 249 , BP 291, BP 303, BP 305, BP 320, BP 338, BP 340, BP 343, BP 347, BP 349, BP 377
Size big vs Size small	BP 2,BP 14,BP 21,BP 22,BP 38,BP 126, BP 155, BP 168, BP 175, BP 237, BP 241, BP 278, BP 279, BP 280, BP 301
Symmetry vs Non-Symmetry	BP 152,BP 22, BP 172, BP 265, BP 269, BP 342
Triangle vs Not Triangle	BP 10, BP 96, BP 97, BP 98, BP 146, BP 151, BP 160, BP 193, BP 194, BP 250 , BP 251, BP 272
Vertical vs Horizontral	BP 5, BP 7 , BP 13, BP 19, BP 65, BP 95, BP 147, BP 258

Figure 4.4: Dataset generated for 20 Concepts.

To create datasets for each concept, I've used Python with important libraries like OpenCV, NumPy, and others. These libraries help me generate images for specific concepts. I've written a code that generates random images by using random.randint function that generates a random number with the help of this I generated random coordinates of shapes and made a dataset for concepts.

The code generates a set number of images for each concept and saves them in a specific path. This way, I've been able to systematically create datasets that have the important features of each concept without any repetition.

4.3 External Datasets Integration:

In this section, I'll discuss that I have downloaded external datasets to enhance the diversity of my dataset. Specifically, I downloaded two datasets from Kaggle. The first dataset contains handwritten English alphabet characters [15]. Since generating a variety of alphabets through Python code was challenging, this dataset was a valuable addition. The second dataset that I downloaded is hand-drawn shapes like rectangles, triangles, and ellipses [16]. By adding these external shapes to my dataset, I have a wider range of variations and complexities to my dataset of Bongard problems. With this approach, I have a dataset that has a lot of variation to learn and later deal with real Bongard problems.

Furthermore, I have written a code to modify the external datasets. These datasets initially consisted of images with white shapes on a black background, with sizes of 28 by 28 pixels, and in JPG format. Using the code, I converted the background to white and the shapes to black. Additionally, I resized the images to dimensions of 100 by 100 pixels and converted them into PNG format. So, all my datasets have the same properties.

4.4 Image Processing and Segmentation:

In this section, I take a step to prepare the Bongard problem images for further analysis. First, I obtained the Bongard problem images from [3] which were originally in GIF format. So, I converted all these images into PNG format. Each Bongard problem image consisted of 12 sub-images that required individual processing.

Thereafter, I wrote a Python code to perform image cropping. This code uses coordinates that I obtained from [17] that crop for all 12 sub-images within each Bongard problem. By doing this I successfully extracted the 12 sub-images from each Bongard problem image. This segmentation process was crucial as it allowed me to crop all sub-images from a bongrad problem so that I could use these images for training and testing purposes.

4.5 Dataset Composition and Diversity:

In this section, I present an overview of my final dataset preparation. For the pre-training of my multi-head CNN model, I generated approximately 45,000 images, for 20 concepts. Additionally, I have taken around 15,000 images from external datasets, so I'll have a variation in my dataset of roughly 61,000 images for the initial model training.

Thereafter, I have 4,692 images for 391 Bongard problems, where each problem contains 12 associated images. Within this dataset, I allocated 10 images for training and reserved 2 images for testing purposes. This dataset I used for transfer learning to check how my model is able to perform on real bongard images after pre-trained on multiple concepts.

4.6 Dataset Preprocessing:

Within this section, I have used a code to get my images ready for training. Here's how it works:

4.6.1 Loading Images:

I've written code to load images from specific folders. These folders contain pictures of different concepts, like "Center vs Not Incenter," "Circle vs Not Circle," and so on. I've made sure that all images are grayscale, which means they are in black and white, to keep things simple. Then, I convert the images into arrays that a computer can understand.

4.6.2 Normalization:

To make the data more manageable for the model, I adjust the values in the arrays. Instead of using numbers from 0 to 255, I make them between 0 and 1. This way, the model can work with the data more effectively.

4.6.3 Creating Labels:

Each image belongs to a specific concept class. For example, an image might be related to "Center vs Not Incenter," or "Circle vs Not Circle." To help the model understand this, I create labels. If an image belongs to the "Center" class, its label is 0. If it's from the "Not Incenter" class, its label is 1.

4.6.4 Splitting Data:

Now, I divide the data into two parts: training and testing. The training data helps the model learn, while the testing data helps us check how well the model is doing. I put aside about 20 percent of the images for testing and used the rest for training.

I repeat this process for different concepts, creating separate sets of training and testing data for each. This prepares the data so that when I start training the model, it already knows what each image means and how to work with them. The whole process takes some time, but it's crucial to make sure the data is ready for the model to learn from.

4.7 Dataset Availability and Access:

The dataset used in this thesis, along with the code used for its generation, is available on my GitHub. Anyone can access the dataset that contains the images for 20 concepts (generated and from external datasets), and cropped images for Bongard problems, also The code used for generating and preprocessing the dataset is available, through the following link for all Thesis files coding and Dataset in the zip: **Click here**

Chapter 5

Methodology

In this chapter, My goal is to make a model that can understand different concepts and solve problems related to them. I came up with 20 different concepts that belong to the Bongard problems, and I made a dataset for each concept as I mentioned in my data chapter. However, to make a model that can learn different concepts there are a few ways. The first approach is to make multiple binary classification models, each for one type of concept, and solve Bongard problems that belong to those concepts. The second approach is: Making a Multi-Head CNN which allows me to handle multiple classification tasks, where each head works as a binary classification task. After discussion with my professor, I realized that I would go for approach second because this multi-head CNN lets me share the lower layers of the neural network across all tasks, capturing common features and relationships among different concepts efficiently that learn shared patterns among tasks, which can improve its ability to generalize and make accurate predictions. However, in the first approach, if we get any Bongard problem that doesn't belong to those concepts then again we need to make one classifier that can handle those Bongard problems, which is not a good approach where we always need to modify our model for future problems.

5.1 Initial Model Development

Building the foundation of my model, I began by creating an 'Input' layer that takes in pictures. Each picture is 100x100 pixels in size and is in black and white (gray) so the input size is (100,100,1). Next, I used 'Conv2D' layers to analyze the pictures. These layers help the model to learn important features and patterns in the images. I used 32 filters in each 'Conv2D' layer, and each filter looks at a small 3x3 section of the image. After each 'Conv2D' layer, I added 'BatchNormalization' to keep things balanced and 'MaxPooling2D' to shrink the image and focus on the important parts. I repeated this process three times to really dive into the details of the pictures. After all the 'Conv2D' and 'Pooling' layers, I flattened the results into a single row of numbers using 'Flatten'. This helps my model to understand the patterns better.

Now, I added a 'Common Layer' with 'Dense' layers. These layers are like the brain of my model, helping it understand the meaning of the pictures. I used 128 neurons in the first 'Dense' layer and 10 neurons in the second one. The choice of 10 neurons in the second layer is intentional because I plan to use this model for transfer learning. In that scenario, I'll have a few images to train my model so the best approach would be to train less weight when we have fewer data to be trained.

But these Individual Heads performed really very well, It seems like they are focused on the concept that they have trained. For example the center of an image or deciding if it's a circle or not.

I trained each 'Individual Head' to give me a '0' or '1' answer for its specific task, and I used the 'sigmoid' activation function to do that. Thereafter, I combined all these 'Individual Heads' into one big model using the 'Model' function. To teach my model, I used the 'optimizer' called 'Adam.' which is also known for learning rate. For each 'Individual Head,' I chose 'binary_crossentropy' as a way for my model to learn from its mistakes and get better at its tasks.

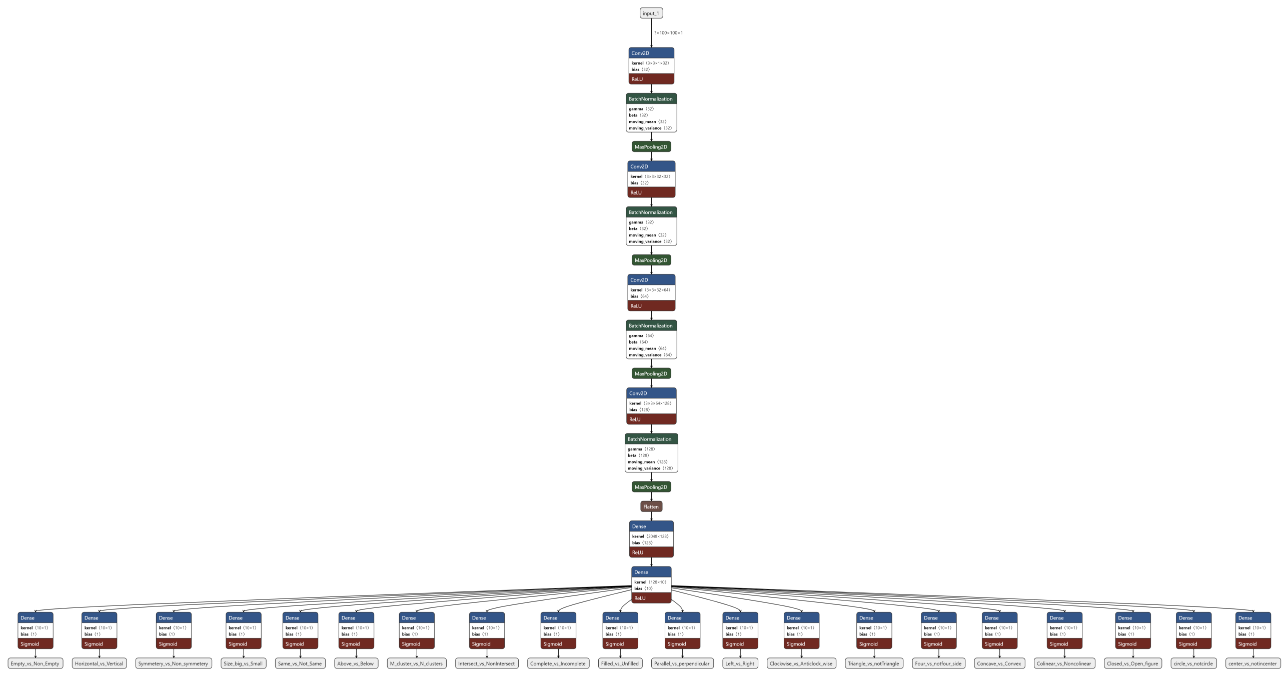


Figure 5.1: Model Architecture
[18]

After developing my model, I began the training process, Where I trained my model with batches of data.

5.2 Transition to Batched Training

In this section, I'll discuss my idea of passing data into a model, I decided to train my model using batches of data, which means groups of images together. To make this work, I put all the training images from different concepts into one batch and made sure that the data was well-organized, while also preparing labels for each image. I organized the labels in a special way so they matched the images correctly. For some images, I set the labels to -1, which helped my model to ignore those images when I passed the first batch. I achieve this by defining a custom loss function in my model, I've developed a custom loss function called masked_binary_crossentropy to improve the training of my model. This function can handle situations where certain labels are marked as -1, indicating that they need to be ignored during training.

After setting up the data and labels, I trained my model using these batches. This change really improved things. The training loss got better, and even the testing accuracies showed improvement. This made me realize that training the model with batches of data was a really smart move. It's like giving the model organized portions of information to learn from.

5.3 Final Model Development

After seeing results from the previous approach which performed well when we trained the model in batches, So, I decided to use this approach and train my model for all my 20 concepts by using this approach.

After I trained my model, I achieved less training loss and good test accuracies, So, I decided to use this model for transfer learning to solve Bongard problems. After training my model where my model learns multiple concepts I freeze those layers weights till the dense_1 layer (which has 128 neurons), and remove all the heads (20 concepts) and just add one new head named 'Bongard' as shown in figure 5.2, where in this head it has only 1 neuron with sigmoid activation function that gives output between 0 and 1 where 0 represent the left side (Set A) and 1 represent the right side (Set B). The idea to train and test the Bongard problem is like this, I can pass 10 random Bongard images out of 12 with their label which represent 0 for the left side and 1 for the right side, and test the rest 2 images and check does my model is able to predict the correct labels or not.

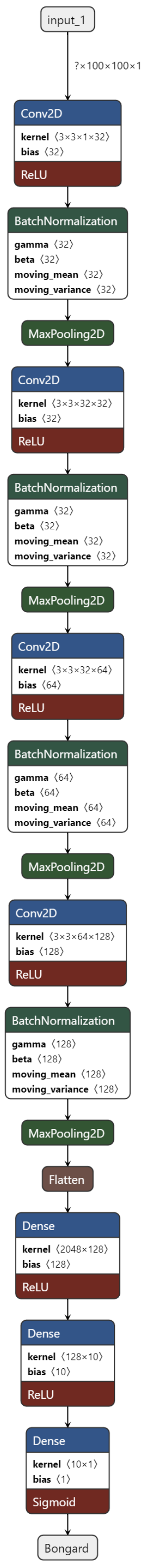


Figure 5.2: Final Model
[18]

Also, while training and testing each Bongard problem, I used the approach cold-start model, where the previous layers were frozen as it was, but every time I passed a new Bongard problem, the dense layer with 10 neurons got reset means it didn't have the memory what it learn from previous bongard problem. I was able to achieve this by making a function called reset_bongard_layer_weights() this function selects the particular Bongard layer and sets the weights by using glorot_uniform() function inside a for loop so that it can set all the weights. So while training and testing the Bongard problem in a loop, I called this function so that every time the next bongard problem came it reset the weights from which learned from the previous problem.

Chapter 6

Experiments and Results

In this chapter, I will discuss the experiments that I did when making a multi-head CNN (as described in my methodology section) and their corresponding results.

6.1 Experiment on Baseline Model:

As discussed in the methodology section about my multi-head CNN, I started training my model where 'X' represents the input data for each concept, and 'Y' holds the labels for each concept. I grouped them in a way (passed in a dictionary) that matched each concept's training data with its respective labels as shown below.

```
model.fit(X[i], {output_names[i]: Y[i]}, epochs=1)
```

To organize the training, I used 'output_names' to specify the names of the different concepts I'm trying to teach my model. This helps the model understand which concept it's focusing on.

I set the number of 'epochs' to 10, which means the model will go through the training data 10 times to learn better. I also defined 'num_heads' as 20, indicating that I'm training all 20 concepts simultaneously.

For each epoch and each head, I used a loop to go through the training process. The model trains on the input data 'X' and its corresponding labels 'Y' for that specific concept's head. After each epoch, I recorded the training loss, which tells me how well the model is learning.

Additionally, once I finished training my model, I looked at the average loss for each epoch. I noticed that instead of going down, the loss was actually going up. Moreover, I checked the test accuracies but they were not as high as I had hoped for and I was struggling to learn the concepts effectively as shown in figures 6.1 and 6.2. So, I decided to investigate this further.

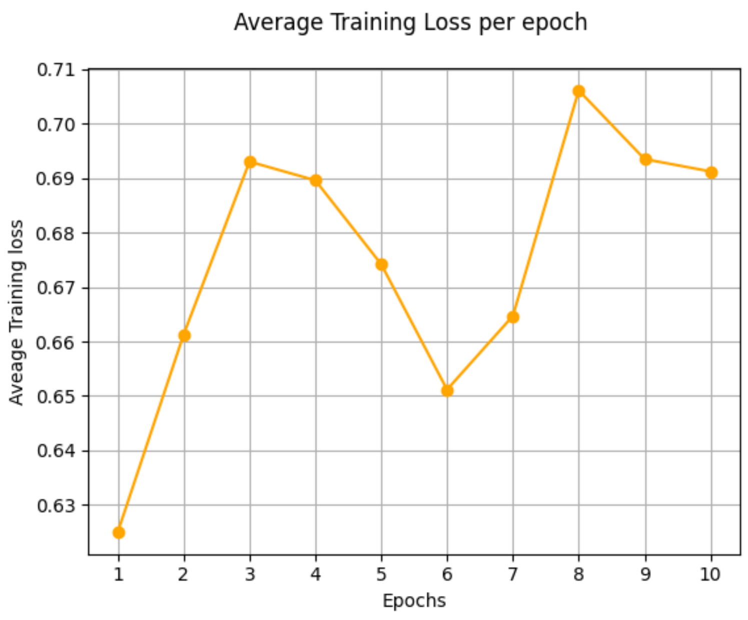


Figure 6.1: Training Loss of Baseline Model

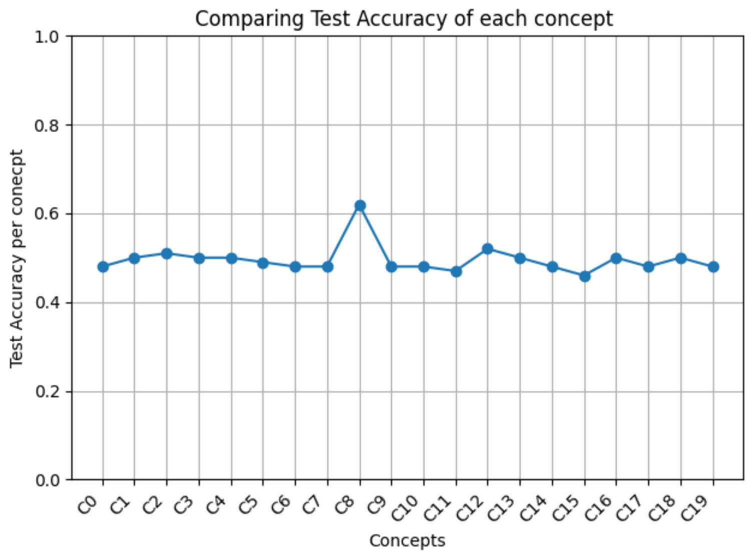


Figure 6.2: Testing accuracy of Baseline Model

6.2 Experiment 1: Individual Learning with cold-start model

In this step, I went step by step to figure out what was causing any problems in my model. So, I trained my model for each concept separately. I wanted to check the accuracy of each concept in a fresh way, so I started from scratch, like starting a new task. I created a new model for individual runs for each concept (Cold-start). Then, for each concept, I trained the model on its corresponding dataset for 10 epochs. This way, I could see how well the model learned each concept individually.

While training each model for each concept, I also added their accuracies to a list. Interestingly, even though the training loss improved, I noticed that the testing accuracy for a few concepts wasn't as good as I hoped. This mismatch between training loss and testing accuracy indicated that my model wasn't generalizing well to new, unseen data for these specific concepts. However, The results are shown in figure 6.3, and 6.4. They show that concepts Such as 4, 7, 9, and 13 were not performing well in terms of test accuracy.

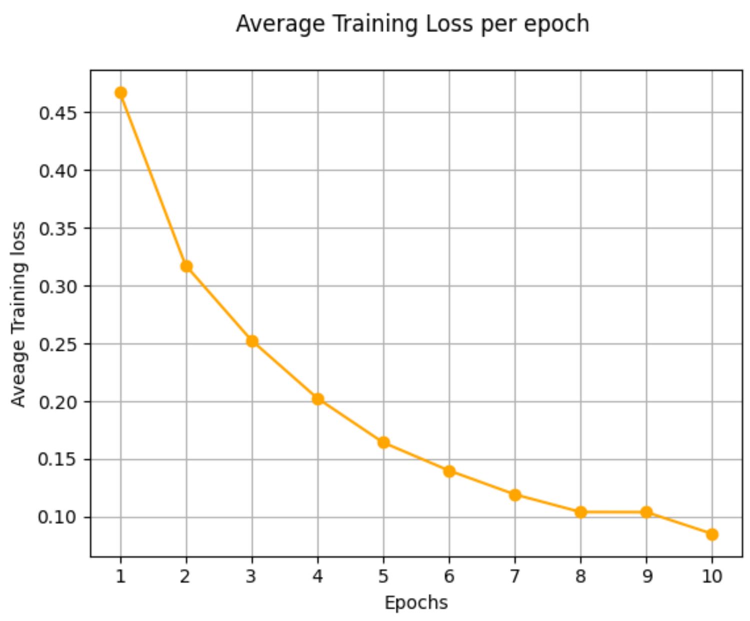


Figure 6.3: Average Training Loss from Experiment 1 (Individual Learning with cold-start model)

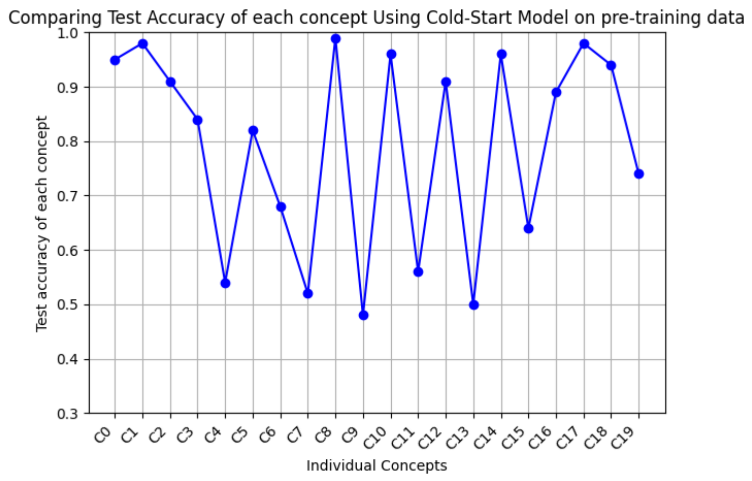


Figure 6.4: Average Test Accuracy from Experiment 1 (Individual Learning with cold-start model)

6.3 Experiment 2: Individual Learning with warm-start model

In my Second experiment, I took a different approach. In this approach instead of creating a separate model for each classifier, I created a single model (Warm-Start model) So that it continues training from the weights that were found in previous training. I saw how well it did on tests and how well the training loss in each concept. I did this for all 20 concepts, keeping track of their progress. The interesting part is that I didn't start from scratch each time. I kept learning from the previous concepts, like using a warm start.

But, even though the model's learning improved (Training loss improving per epoch) as it went on, But again I noticed that it still didn't do well on the few tests set. That gave me a clear idea of what parts needed more attention and fixing. The results are shown in figure 6.5 and 6.6. They show that concepts that I struggled with in Experiment 1, specifically Concepts 4, 7, 9, 14, and 19, again gave bad test accuracy with the warm start approach as well.

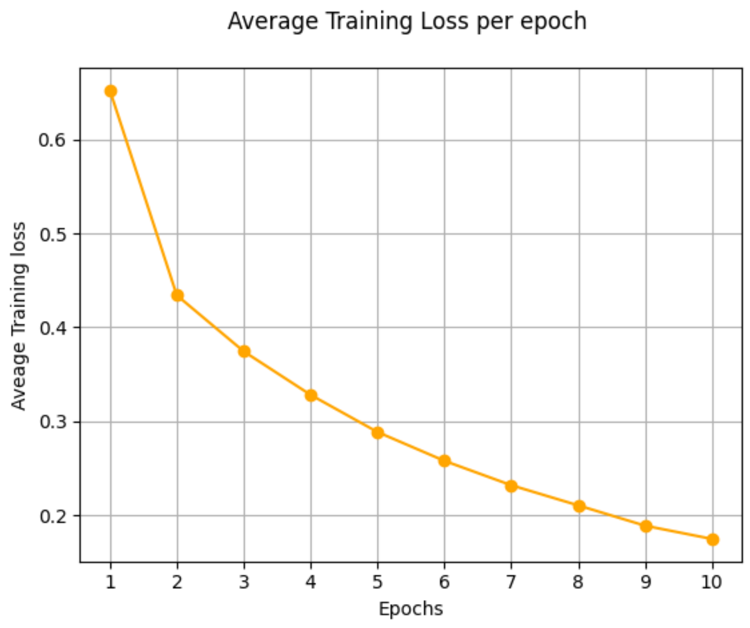


Figure 6.5: Average Training Loss from Experiment 2 (Individual Learning with Warm-start Model)

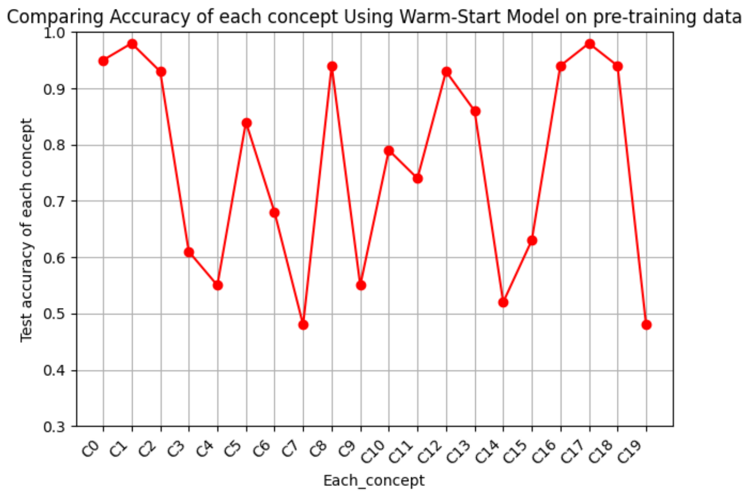


Figure 6.6: Average Test Accuracy from Experiment 2 (Individual Learning with Warm-start Model)

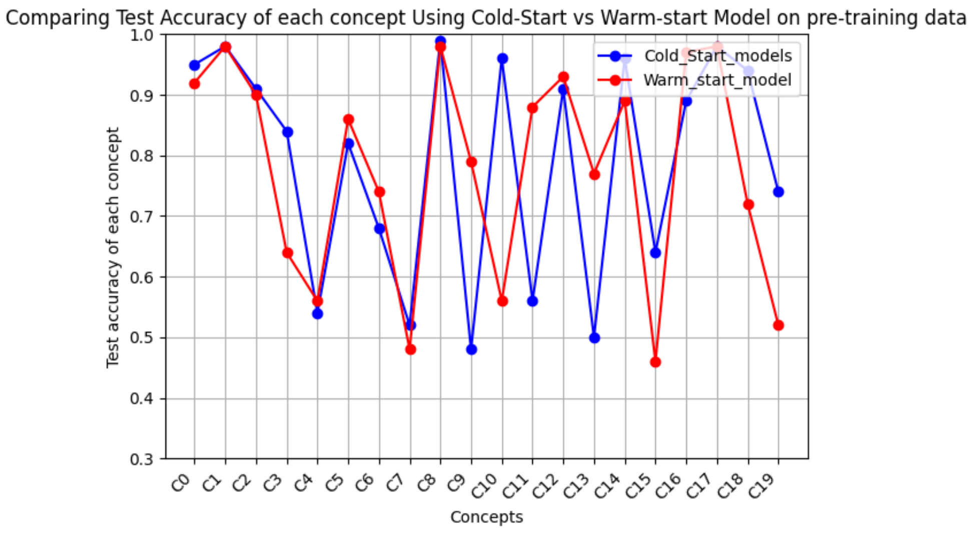


Figure 6.7: Experiment 1 and Experiment 2 Test accuracy comparison (Cold-start model vs Warm-start model)

6.4 Experiment 3: Sequential Subset Learning with cold-start model

In my third experiment, I wanted to find out why my main model wasn't learning well. Even though individual concepts seemed almost fine, my main model struggled. So, I tried training the model with smaller groups of concepts (Sequential subsets), starting with one and adding more gradually. I used a fresh new model (Cold-start) for individual run for each Sequential subset to check whether each subset is performing well or not.

For example, I trained a new model just for concept 0. Then, I made a new model for concept 0 and concept 1, and so on, until I used all 20 concepts. I used to train all concepts(heads) that are inside the subset in each epoch, and I did till 10 epochs. By using the below logic.

```
"for epoch in epochs:
```

```
for i in num_heads:
    Model_history=model.fit(X[i], {output_names[i]: Y[i]}, epochs=1)"
```

After I trained my all models for all sequential subsets I noticed my training loss was not improving as subsets were increasing, instead of decreasing the training loss its increasing and there were a few concepts whose test accuracies were not good in every subset. This made it clear that training all the concepts together at once was causing complexity issues for the model. No matter how I grouped them, the results didn't get better as shown in figures 6.8 and 6.9. Moreover, there were some concepts that again performed badly they are: Concepts: 3,4,7,9,14 and 19.

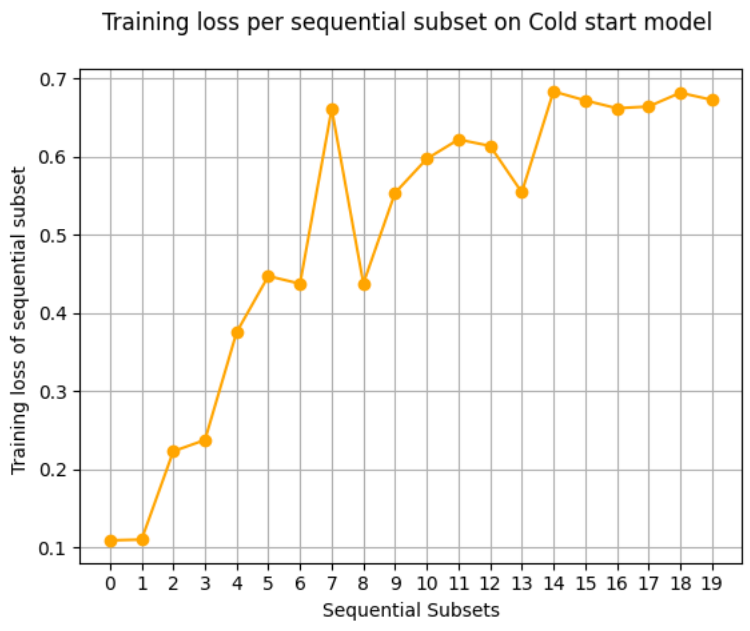


Figure 6.8: Avergae Training Loss per sequential subset form Experiment 3 (Sequential Subset Learning with cold-start model)

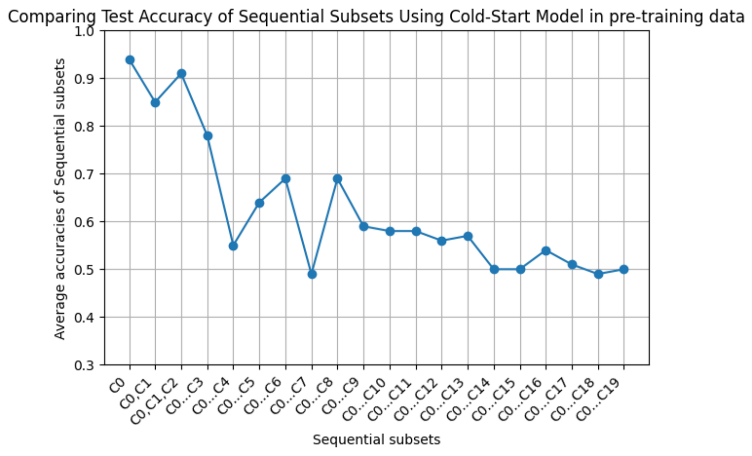


Figure 6.9: Avergae Test accuracy form Experiment 3 (Sequential Subset Learning with cold-start model)

6.5 Experiment 4: Sequential Subset Learning with warm-start model

In the fourth experiment phase, This time, I focused on using a warm start model where it continues training from the weights that were found in previous training. This means that instead of starting with a completely fresh model for each subset of concepts, I built a model that has existing knowledge gained from previous subsets. This warm start approach can be more effective in training the model.

While training a model, I kept an eye on the training loss and how well the model was doing on new tests. Training loss tells that the model is getting better at understanding the concepts, while test accuracy shows how good it is at guessing new things.

Things were improving, both the training loss and test accuracy were getting better and better as the model learned from each subset of concepts. This was totally different from when I tried this with the cold start model (experiment 3) in the last phase. That time, the training loss didn't improve. However, the similarity that I noticed in experiment 3 and experiment 4 is that there were a few concepts that gave bad test accuracy in every subset. The results are shown in figure 6.10 and 6.11. In this case, I realized that training loss started decreasing as we increased the number of concepts in the subset. Unfortunately, testing accuracy was not as good as expected. However, still some concepts that didn't perform well in subsets: concepts 3, 4, 7, 9, and 15.

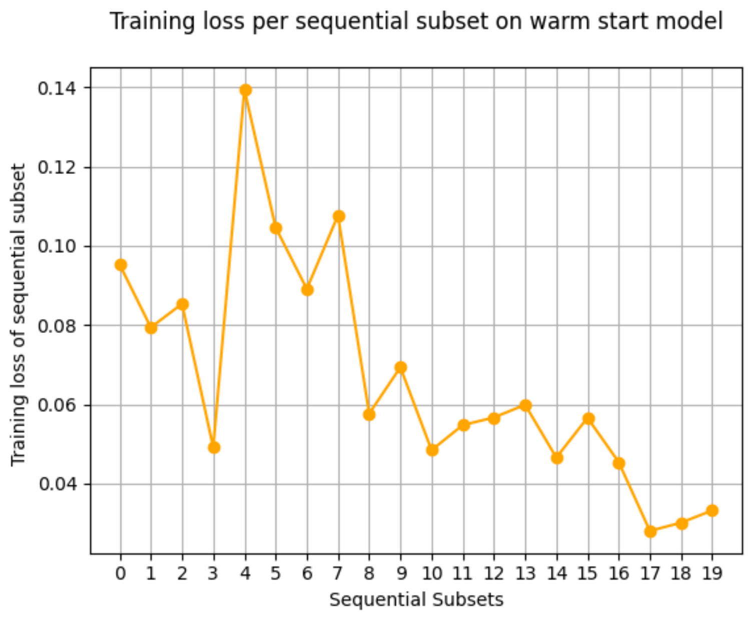


Figure 6.10: Avergae Training Loss per sequential subset form Experiment 4 (Sequential Subset Learning with Warm-start Model)

6.5 Experiment 4: Sequential Subset Learning with warm-start model

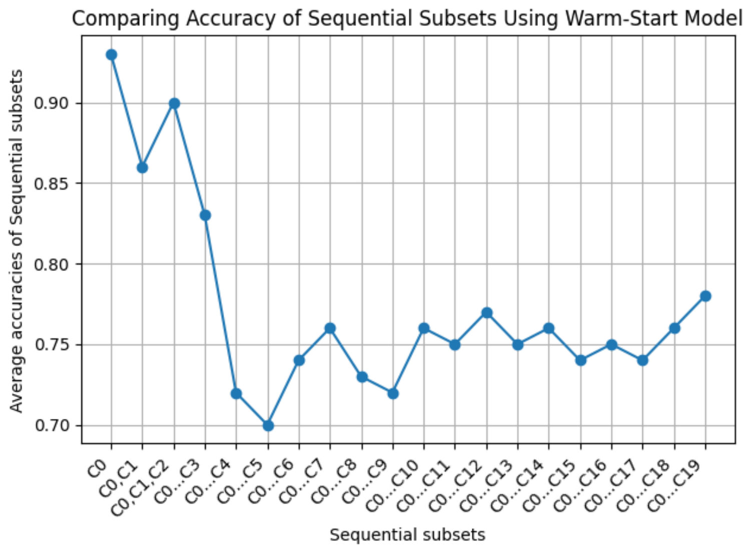


Figure 6.11: Avergae Test accuracy form Experiment 4 (Sequential Subset Learning with Warm-start Model)

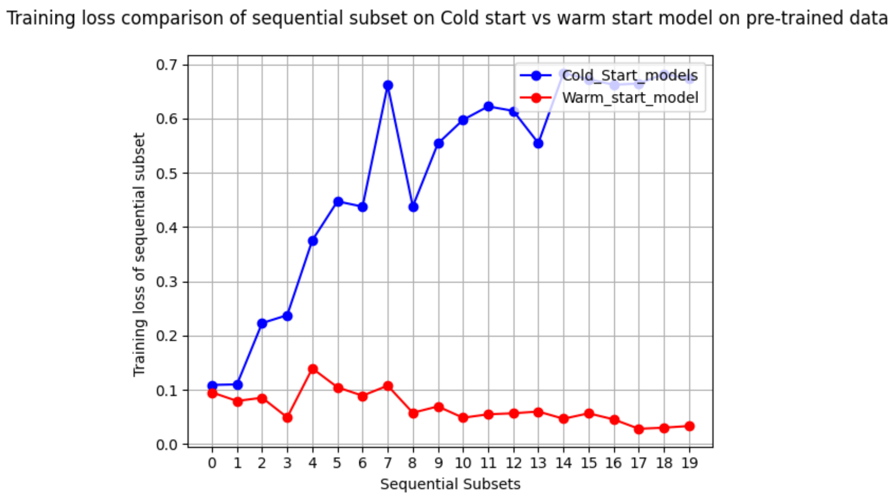


Figure 6.12: Comparison of Experiment 3 and 4 Training loss (Cold-start vs Warm start Sequential Subset Model)

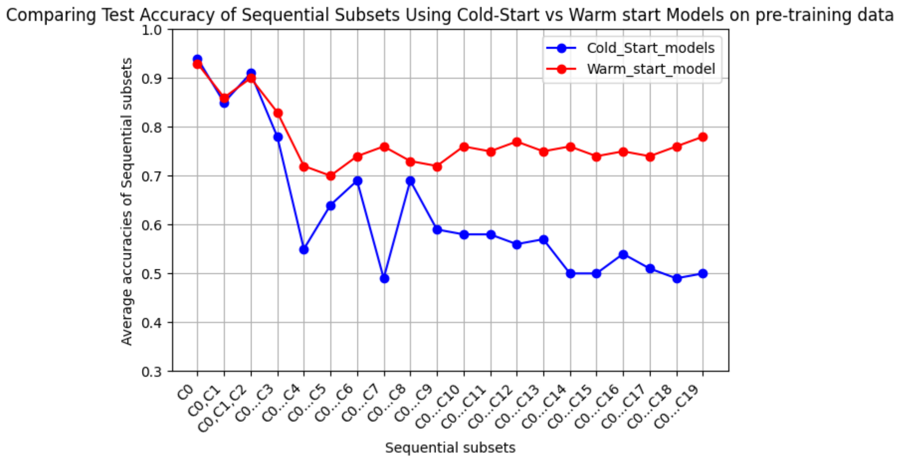


Figure 6.13: Comparison of Experiment 3 and 4 Test Accuracies (Cold-start vs Warm start Sequential Subset Model)

6.6 Experiment 5: Addressing Challenge Concepts

From the above experiments, the one thing that is common is that there were few concepts that performed badly either I did individual concept training or in subsets training, so this time I went back to my main model, and for those concepts that performed badly in the test set, I increase the epoch for those concepts only by using if statement if those concepts come under the loop then for those concepts model will take extra epochs to train the model, as shown below:

```
for epoch in epochs: #epoch =10
    for i in num_heads: #num_head=20
        if i==3 or i==4 or i==7 or i==9 or i==15:
            Model_history=model.fit(X[i], {output_names[i]: Y[i]}, epochs=10)
        else:
```

```
Model_history=model.fit(X[i], {output_names[i]: Y[i]}, epochs=1)"
```

After training my main model with this modification and with the hope that this time everything will go fine and I'll get better training loss and test accuracies, this time training loss was improved as expected but unfortunately, testing accuracies were not as good as expected. The results are shown in figures 6.14 and 6.15. However, my training loss improved as expected but my test accuracies weren't improved as I hoped.

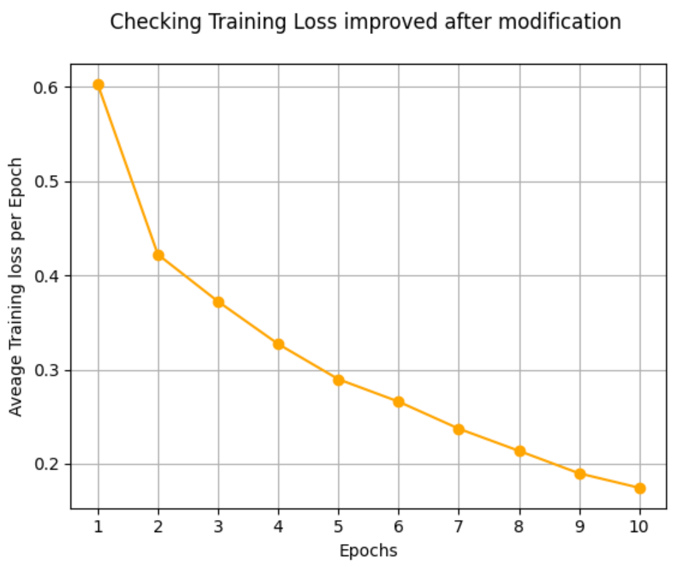


Figure 6.14: Average Training Loss form Experiment 5 (Addressing the Challenge Concept)

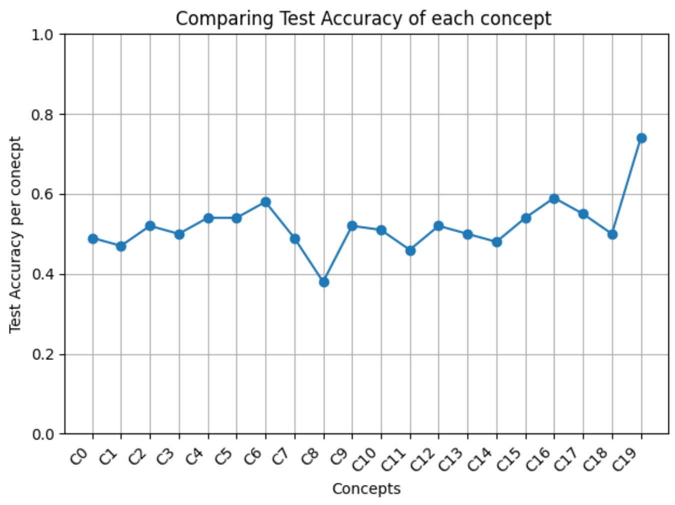


Figure 6.15: Average Test Accuracy form Experiment 5 (Addressing the Challenge Concept)

6.7 Learning:

After carefully analyzing all the experiments and discussing them with my professor we realized that our model has started learning but is unable to give good performance on unseen data. This made us realize that instead of training the model in a loop we have to figure out some other approach. However, we came up with a different approach that is passing data in batches which means we can make a dictionary for labels and concatenate all training images in an array for training data which we already discussed in the Methodology section.

6.8 Results from final model

As discussed in the methodology section after experiments I realized to train my model in batches and below are the results that I received when I checked the performance of my final model.

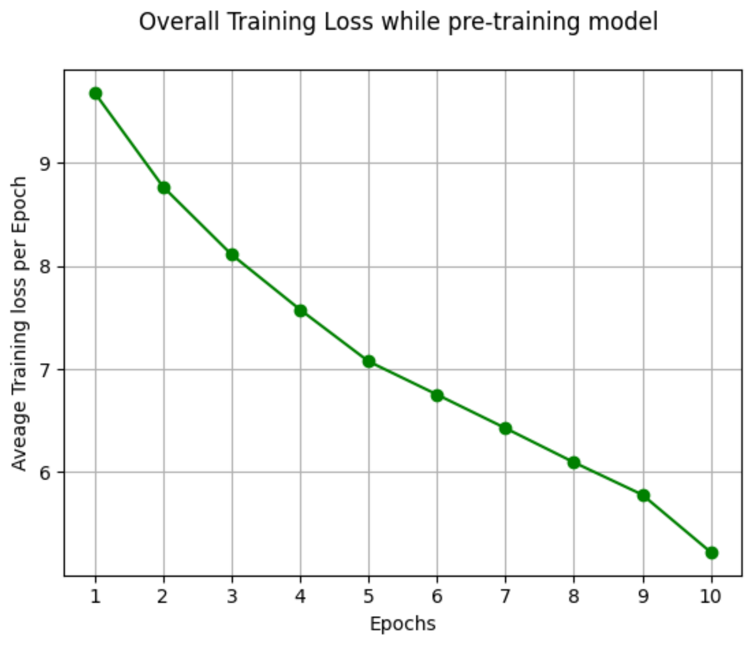


Figure 6.16: Training loss of final model (Batch training)

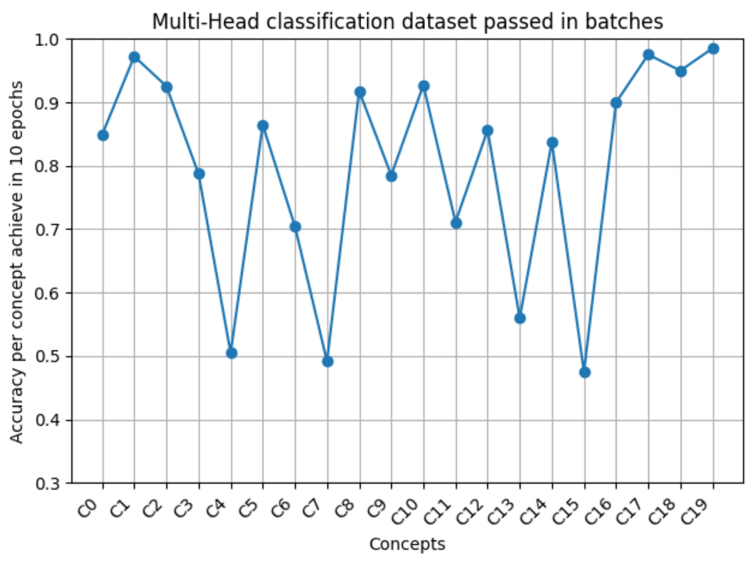


Figure 6.17: Average Test Accuracy of final model (Batch training)

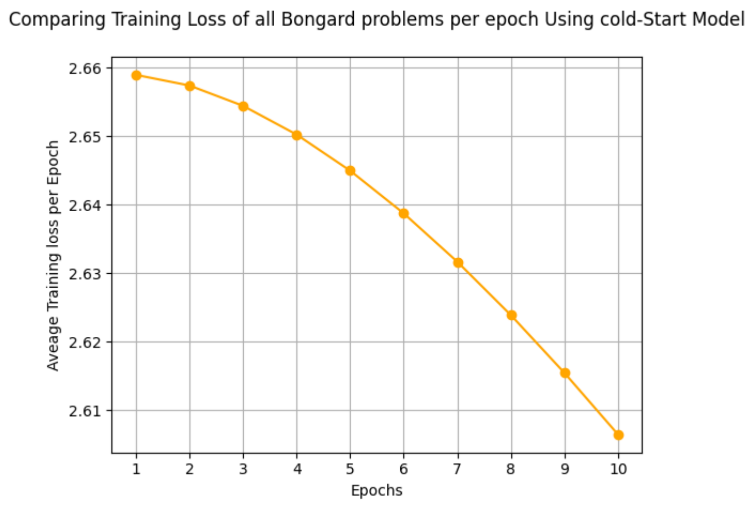


Figure 6.18: Average Training loss on Bongard problem by using final model

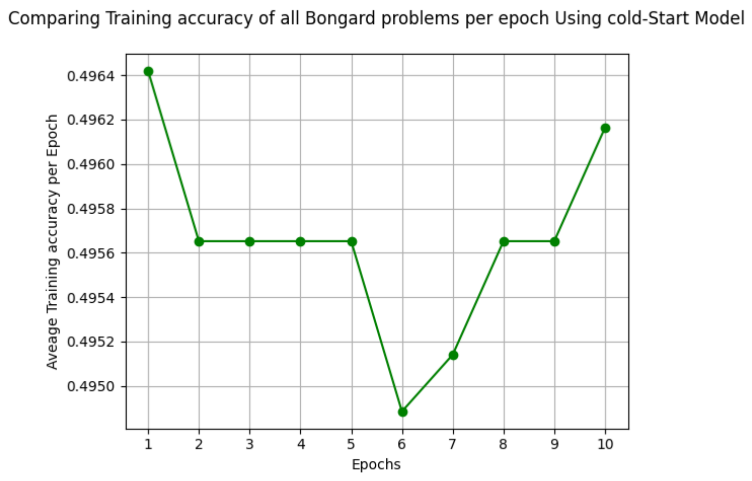


Figure 6.19: Average Training accuracy on Bongard problem by using final model

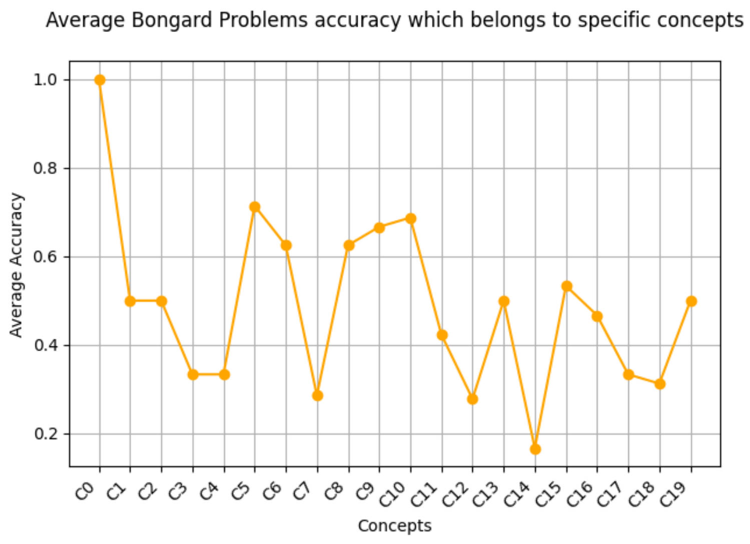


Figure 6.20: Average Test accuracy of Bongard problems with their respective concepts

6.9 Discussion of Results in Relation to Research Questions

Answer RQ1) When I trained my model by focusing on each concept separately (in experiment 1 and experiment 2), I observed something interesting. My model was actually able to learn several concepts quite well, whether I used a cold start or a warm start approach as shown in the figures 6.3 (Individual Head), 6.5(Individual Head) and 6.16 (Multi-head Training loss from the final model). Interestingly, with the warm start, I noticed a slight improvement in performance

Moreover, when I use the final approach where I pass the data in batches, my model is able to learn all concepts very efficiently as I have seen in average training loss which has been reduced continuously as shown in figure 6.18. Also, my model began to pick up on patterns in the Bongard problems. While the reduction in training loss wasn't huge, it was definitely noticeable, and this improvement showed that my model was learning the concepts more effectively. Moreover, It is crystal clear that the model is able to learn different concepts because these figures represent that training loss is decreasing continuously. However, the learning pattern is still challenging for the model because the training loss isn't huge.

Answer RQ2) When I checked my final model to analyze the test accuracies for all concepts, approximately I received 75 percent average accuracy. On the other hand, when I checked the accuracy for Bongard problems I only received around 50 percent and my model is able to solve around 100 Bongard problems out of 391. This indicates that my model demonstrates proficiency in differentiating between concepts, but it faces challenges when it comes to distinguishing Bongard problems effectively. However, according to the graphs 6.4, 6.6 and 6.17, my model is able to tell the difference between concepts and Bongard Problems because it performs well in test accuracies. However, according to figure 6.19 my model doesn't perform well to tell the difference between Bongard Problems because training accuracy is not good.

Answer RQ3) During my analysis of the different stages of experimenting (experiment 1, experiment 2, experiment 3, and experiment 4), I observed some interesting patterns in my model's performance. When I trained the model to learn individual concepts using both cold start and warm start approaches (experiment 1 and experiment 2), I noticed that there wasn't a significant difference in performance between the two methods as shown in figure 6.7.

However, things took a turn when I experimented with training the model on sequential subsets of concepts, utilizing both cold and warm start models (experiment 3 and experiment 4). In this scenario, I found that my model performed better learning when using a warm start approach. It was able to grasp multiple concepts, leading to improved training loss and testing accuracies. However, the opposite happened with a cold start model—there was trouble in learning, with training loss it increased instead of decreasing, and testing accuracies remained around 50 percent. This observation tells that the warm start model approach is better in this scenario. Therefore, according to figures 6.12 and 6.13 it is clearly observed that there is a huge difference when we train our model by sequential subsets in cold-start vs warm-start.

Answer RQ4) After my experiments on experimenting with the model, I realized that performance is better when I train individual concepts as compared to multiple concepts, I realized this in experiment 3 and experiment 4 where when I increase the concepts in subsets then my model performance got a slight decrease in terms of both training loss and in testing accuracy in both situations either I do warm start model or cold start. These results indicate that the model struggles to manage high levels of complexity all at once. Moreover, These figures 6.7 6.13 6.12 also indicate that performance on individual concepts is slightly better as compared to multiple concepts.

Answer RQ5) Well, the pre-trained dataset which I generated for 20 concepts, belongs to 148 Bongard problems out of 391. However, I checked the accuracy of the Bongard problems that belong to my 20 concepts as shown in figure 6.20 and I received around the same accuracy, that I received earlier around 50 percent on overall Bongard problems. But a few concepts such as "center vs. not in the center" Bongard problems which come under this concept achieve 100 percent accuracy, and concepts such as "Triangle_vs_Not_Triangle", "Four_sides_vs_not_four_sides", "Left_vs_Right ", "Parallel_vs_Perpendicular" and "filled_vs_unfilled" has achieved more than 60 percent accuracy. However, According to the graph 6.16, It represents that our model has the ability to learn multi-concepts. So I believe that yes this pre-training can help to solve Bongard's problems.

6.10 Summing Up

This part brings everything together and gives a solid summary of what I have discovered and its importance.

I began by presenting my experiments with graphs. Those graphs gave a first look at how well my model was doing with different things.

The "Discussion of Results in Relation to Research Questions" section went deeper into the results and how they connect to our research questions. I found clear links between what I saw in the graphs and the model's ability to learn, understand differences, and adapt. This detailed analysis shows my model's strengths and where it can do even better. This journey helps us to see what my model can do and where it can take us in the future.

Chapter 7

Conclusion

In conclusion, my research journey ”Solving Bongard Problems with Conceptual Transfer Learning via the Multi-Head CNN” has shown good performance. By generating a dataset for 20 concepts, My goal was to train a model so that it is capable of classifying these concepts, so it can perform well in solving bongard problems. While the achieved accuracy is around 50 percent only, I believe there might be other things to improve.

The results suggest that increasing the variation within the dataset can help the model to learn new patterns, and also instead of 20 concepts it might be a good choice to increase more concepts so that our model can be able to learn multiple concepts with variations. It’s clear that the complexity of Bongard’s problems needs various visual patterns and relationships.

So I’m not stopping here. I believe that if I keep fine-tuning my multi-head CNN thing, and try some new training tricks, and maybe I might just push that accuracy up a notch.

References

[1] Wikipedia contributors. (Year of last update) Bongard problem. [Online]. Available: https://en.wikipedia.org/wiki/Bongard_problem 1

[2] H. Foundalis. (n.d.) Dissertations and research. [Online]. Available: https://www.foundalis.com/res/diss_research.html 1

[3] D. Foundalis. (Year of Retrieval) Bongard problems solutions. Accessed on Date of Retrieval. [Online]. Available: https://www.foundalis.com/res/bps/bongard_problems_solutions.htm 2, 10, 17, 18, 19, 26

[4] IBM. (n.d.) Artificial intelligence - ibm. [Online]. Available: https://www.ibm.com/topics/artificial-intelligence 5, 6

[5] E. Azizi and L. Zaman, "Deep learning pet identification using face and body," *Sensors*, vol. 14, p. 278, 2014. [Online]. Available: https://www.mdpi.com/2078-2489/14/5/278 6, 7

[6] Tanya. (Year) Data preprocessing and network building in cnn. [Online]. Available: https://towardsdatascience.com/data-preprocessing-and-network-building-in-cnn-15624ef3a28b 7

[7] R. Ynr. (Year) Data science and its role in decision making. [Online]. Available: https://medium.com/@rishani.ynr/data-science-and-its-role-in-decision-making-e3c58b681234 7

[8] H. B. Mishra. (Year) Creating dog versus cat classifier using transfer learning. [Online]. Available: https://medium.com/tesseract-coding/creating-dog-versus-cat-classifier-using-transfer-learning-63cac5a8d3d8 8

[9] N. Donges. (Publication Year) Transfer learning. [Online]. Available: https://builtin.com/data-science/transfer-learning 13

[10] P. R. Ahmed. (2022) What is transfer learning? [Online]. Available: https://www.youtube.com/watch?v=3gyeDlZqWko 13, 14

[11] S. Kharagorgiev. (2018) Solving bongard problems with deep learning. [Online]. Available: https://k10v.github.io/2018/02/25/Solving-Bongard-problems-with-deep-learning/ 14, 15, 16

[12] M. fahy, "Bongard problems and modern neural networks," 2022. 14, 15, 17

[13] B. Adhikari. (2021) Deep learning with python - chapter 7 - 7.2 - keras sequential vs functional api. [Online]. Available: https://www.youtube.com/watch?v=EvGS3VAsG4Y 20

[14] A. Rosebrock. (2018) Keras: Multiple outputs and multiple losses. [Online]. Available: https://pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses/ 20

[15] Mohneesh7. (Year of Retrieval) English alphabets dataset. Accessed on Date of Retrieval. [Online]. Available: https://www.kaggle.com/datasets/mohneesh7/english-alphabets?resource=download 25

[16] Frobert. (Year of Retrieval) Handdrawn shapes dataset (hds). Accessed on Date of Retrieval. [Online]. Available: https://www.kaggle.com/datasets/frobert/handdrawn-shapes-hds-dataset 26

[17] PEKO-STEP. (Year of Retrieval) Online image cropping tool. Accessed on Date of Retrieval. [Online]. Available: https://www.peko-step.com/en/tool/cropimage.html 26

[18] Netron. (2023) Netron: Visualizer for neural network models. [Online]. Available: https://netron.app/ 32, 34