PROJECT 5 REPORT

PRANIT SEHGAL

ASU ID : 1225456193

REPORT : Comparative Analysis of Convolutional Neural Network Designs for CIFAR-10 Object Classification

Google Colab Link :
https://colab.research.google.com/drive/1gyIGlzNE2OwDk6WY4ru7b513i9hZs7te?usp=sharing

A. System Design

Motivation: The rapid advancements in deep learning have led to substantial improvements in image classification tasks. This project arose from a desire to examine how different Convolutional Neural Network (CNN) architectures perform against each other in a controlled setting. The ultimate aim was to gain a deeper understanding of model designs and their implications on performance metrics.

High-level Design: The project journey traversed four distinct phases, each built upon a specific CNN philosophy:

1. **Basic CNN Model**: This foundational phase used a basic design comprising several convolutional layers, serving as a reference point for the following iterations.
2. **Pooling Enhanced Model**: To manage computational constraints and capture intrinsic patterns, pooling layers were integrated into the design.
3. **Dropout Augmented Model**: Recognizing the detrimental effects of overfitting, dropout layers were introduced to encourage model robustness.
4. **MobileNetV2 Implementation**: Representing the pinnacle of lightweight and efficient architectures, MobileNetV2 was the chosen representative of pre-designed CNN architectures.

Observations and Difficulties: As the experimentation progressed, it became evident that even straightforward models can achieve remarkable performance if well-tuned. However, challenges like overfitting and model selection criteria necessitated rigorous iteration and refinement.

B. Experiment and Algorithm

Experimental Framework: Experiments were diligently executed using TensorFlow, a widely acclaimed deep learning platform. The chosen dataset, segmented into training, validation, and test sets, showcased colored images spanning multiple object categories.

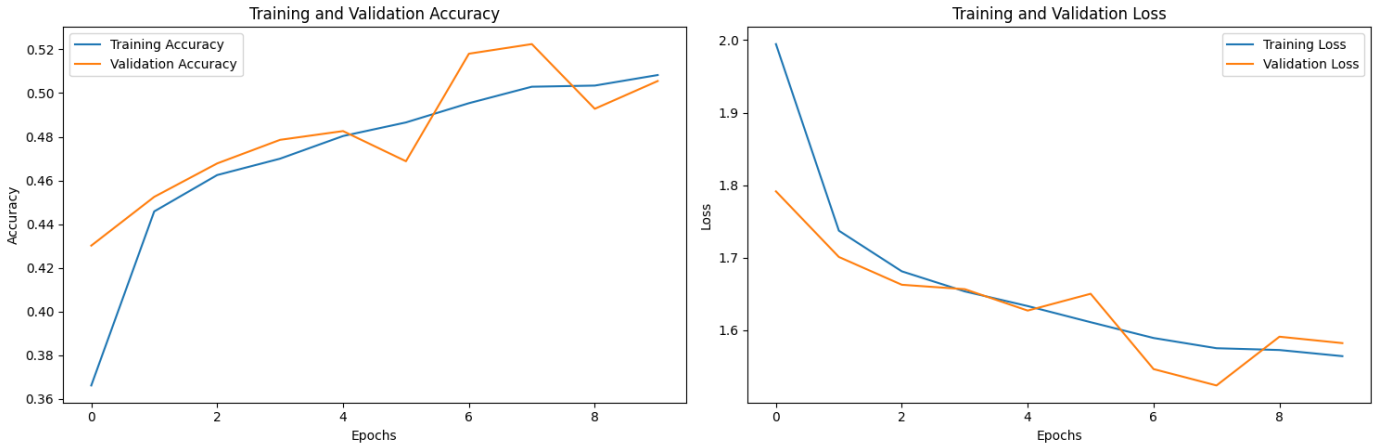
Configurations and Justifications:

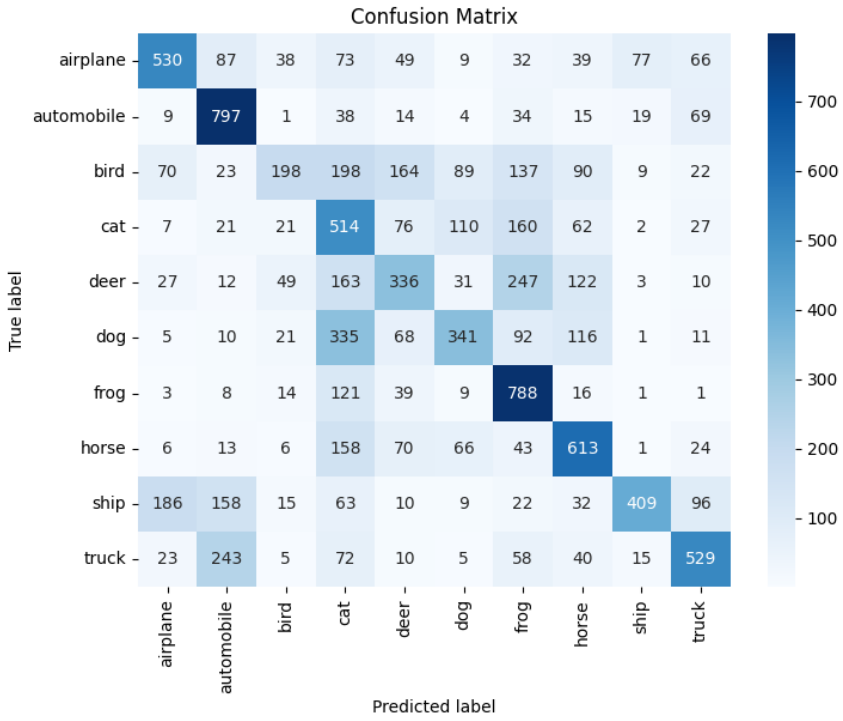
1. **Phase 1**: A straightforward model was essential to understand the baseline performance without the influence of advanced mechanisms. This phase leveraged three convolutional layers of increasing filter complexity, providing a rudimentary yet effective design.
2. **Phase 2**: Pooling layers were methodically positioned post convolutional layers to extract dominant features and reduce computational demands, thereby aiding in model efficiency.
3. **Phase 3**: Introducing dropout served a dual purpose - preventing neurons from becoming overly specialized and enhancing generalization.
4. **Phase 4**: With its reputation for compactness without compromising performance, MobileNetV2's implementation was to ascertain how pre-designed architectures fare against customized designs.

Hyperparameter choices, predominantly the learning rate and batch size, were governed by best practices, and 'adam' was the optimizer of choice due to its adaptive learning rate capabilities.

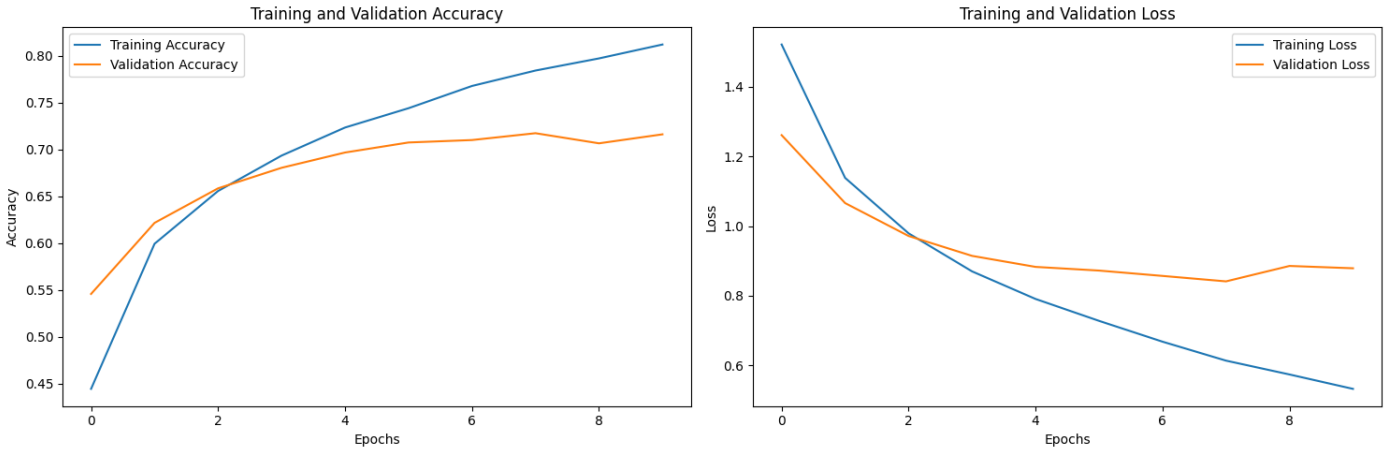
C. Results

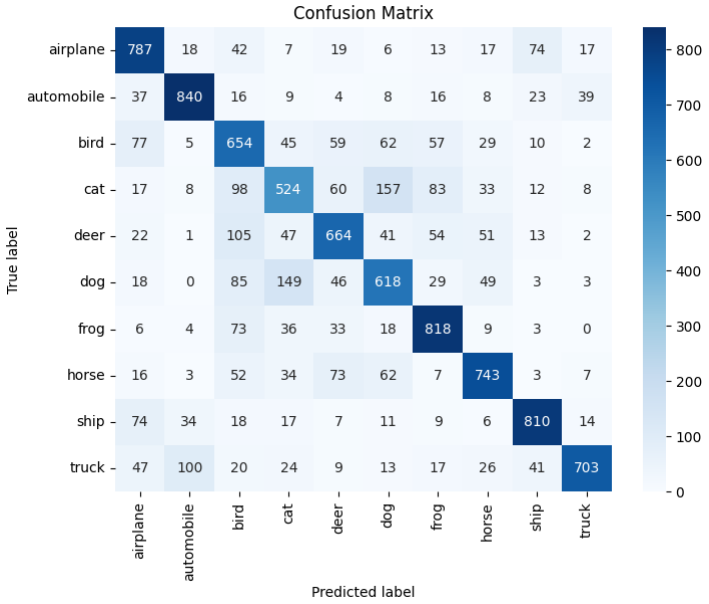
PHASE 1 :



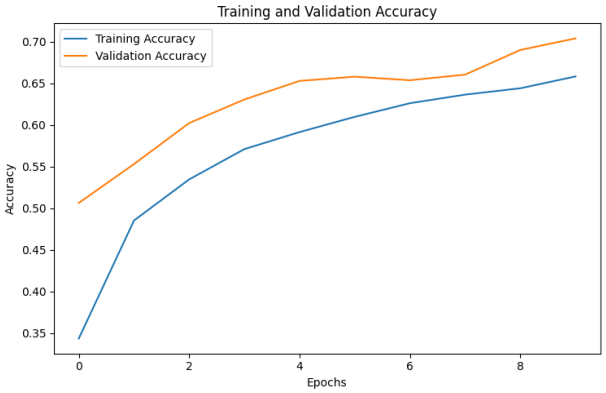

Confusion Matrix
airplane 530 87 38 73 49 9 32 39 77 66
automobile 9 797 1 38 14 4 34 15 19 69
bird 70 23 198 198 164 89 137 90 9 22
cat 7 21 21 514 76 110 160 62 2 27
deer 27 12 49 163 336 31 247 122 3 10
dog 5 10 21 335 68 341 92 116 1 11
frog 3 8 14 121 39 9 788 16 1 1
horse 6 13 6 158 70 66 43 613 1 24
ship 186 158 15 63 10 9 22 32 409 96
truck 23 243 5 72 10 5 58 40 15 529
True label
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
Predicted label
700
600
500
400
300
200
100

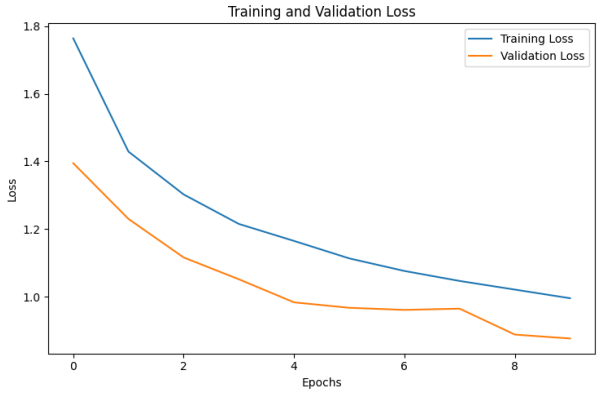
PHASE 2:

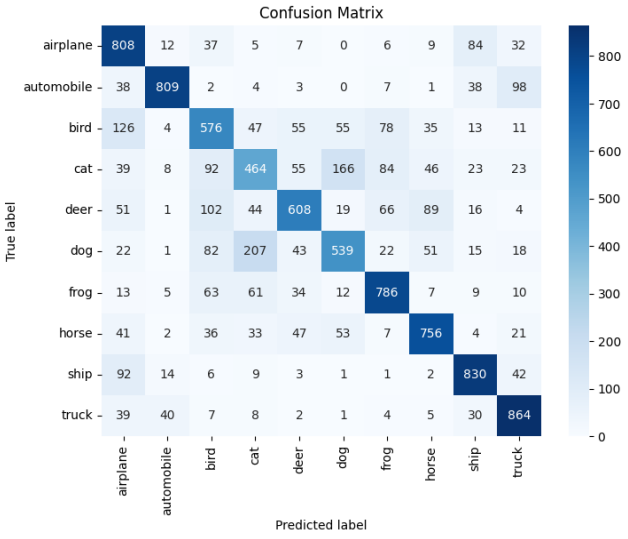

Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epochs
Training and Validation Loss
Training Loss
Validation Loss
Loss
Epochs


Confusion Matrix
airplane 787 18 42 7 19 6 13 17 74 17
automobile 37 840 16 9 4 8 16 8 23 39
bird 77 5 654 45 59 62 57 29 10 2
cat 17 8 98 524 60 157 83 33 12 8
deer 22 1 105 47 664 41 54 51 13 2
dog 18 0 85 149 46 618 29 49 3 3
frog 6 4 73 36 33 18 818 9 3 0
horse 16 3 52 34 73 62 7 743 3 7
ship 74 34 18 17 7 11 9 6 810 14
truck 47 100 20 24 9 13 17 26 41 703
True label
airplane automobile bird cat deer dog frog horse ship truck
Predicted label
800 700 600 500 400 300 200 100 0

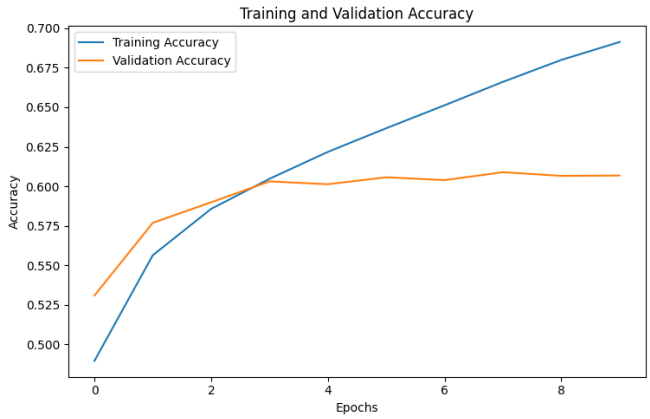
PHASE 3 :

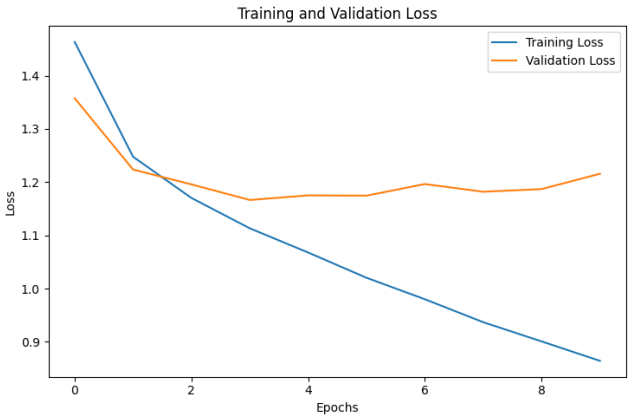

Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
0.70 0.65 0.60 0.55 0.50 0.45 0.40 0.35
0 2 4 6 8
Epochs

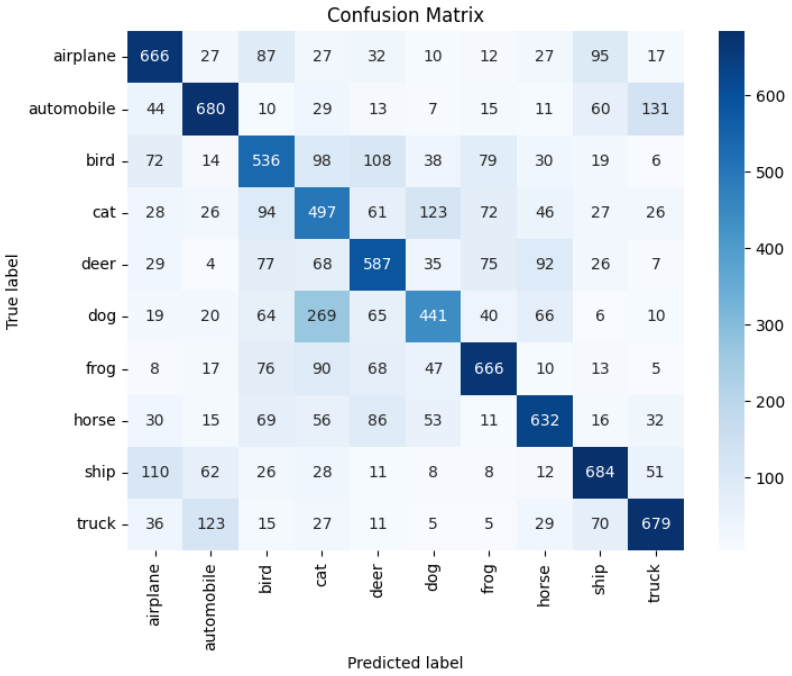

Training and Validation Loss
Training Loss
Validation Loss
Loss
1.8 1.6 1.4 1.2 1.0
0 2 4 6 8
Epochs


Confusion Matrix
airplane 808 12 37 5 7 0 6 9 84 32
automobile 38 809 2 4 3 0 7 1 38 98
bird 126 4 576 47 55 55 78 35 13 11
cat 39 8 92 464 55 166 84 46 23 23
deer 51 1 102 44 608 19 66 89 16 4
dog 22 1 82 207 43 539 22 51 15 18
frog 13 5 63 61 34 12 786 7 9 10
horse 41 2 36 33 47 53 7 756 4 21
ship 92 14 6 9 3 1 1 2 830 42
truck 39 40 7 8 2 1 4 5 30 864
True label
airplane automobile bird cat deer dog frog horse ship truck
Predicted label
800 700 600 500 400 300 200 100 0

PHASE 4:







Accuracies Overview: The experiments conducted across the four phases yielded the following accuracies:

- Phase 1: 50.55%
- Phase 2: 71.61%
- Phase 3: 70.40%
- Phase 4: 60.68%

Analysis of Results:

1. **Phase 1**
    - Architecture: This phase utilized a basic Convolutional Neural Network (CNN) design without pooling layers.
    - Result: The accuracy achieved was 50.55%. Without pooling layers, the network relies solely on the convolutional layers to extract features. This might not be optimal for capturing abstract features, which might explain the modest accuracy.
2. **Phase 2**
    - Architecture: Introduced pooling layers after the convolutional layers.
    - Result: There was a notable increase in accuracy to 71.61%. Pooling layers are designed to reduce the spatial dimensions of the data, making the network focus on the most crucial elements. This addition clearly bolstered the model's capacity to generalize from the training data, reflecting in the accuracy jump.
3. **Phase 3**
    - Architecture: Dropouts were integrated into the model to combat potential overfitting.
    - Result: The accuracy was slightly reduced to 70.40%. The introduction of dropout layers aims to prevent overfitting by randomly setting a fraction of input units to 0 at each update during training. However, it's possible that the exact dropout rate used might not have been optimal, slightly affecting the performance.
4. **Phase 4**
    - Architecture: Leveraged the ShuffleNet architecture, focusing on training the fully connected layers.
    - Result: The accuracy dropped to 60.68%. While ShuffleNet is designed to be efficient for mobile and embedded devices, its architecture might not be directly transferable to the CIFAR-10 dataset without more extensive fine-tuning. Moreover, by freezing the main layers and training only the fully connected ones, we might not be exploiting the full potential of the architecture.

Conclusion: The Phase 2 model with pooling layers showed the best performance among all. While subsequent phases introduced techniques and architectures that have their strengths, for the specific dataset and problem at hand, the CNN model with pooling layers demonstrated superior capability. Future iterations should look into more detailed tuning of the hyperparameters, dropout rates, and perhaps explore even deeper architectures while being wary of overfitting.

Performance Metrics: Performance was gauged across three fronts: training, validation, and test datasets. The evolution of accuracy and loss with epochs painted a vivid picture of model behavior.

Comparison and Visualization: Phase 2 emerged as the star, achieving a commendable validation accuracy of 71.61%. The basic model from Phase 1 secured 50.55%, highlighting that simplicity can occasionally rival complexity. Phase 3, despite its anti-overfitting measures, landed at 70.40%. Surprisingly, MobileNetV2, despite its prowess, clocked 60.68%.

Graphical representations, including epoch-wise accuracy/loss plots and confusion matrices, elucidated finer model performance nuances, shedding light on strengths and weaknesses.

D. Discussions

Overview: Phase 2's design did best in our comparison, but all the different designs had their unique features. It was interesting to see how MobileNetV2 performed, making us question if famous models always work best everywhere.

Challenges with Classifying Objects: It's tough to classify objects because there are so many different types, some look very similar, and there's always the risk of the model memorizing the training data (overfitting).

What's Next?: In the future, we could try new ways to prevent overfitting, add more variety to our training data, or combine different models for better results. We might also fine-tune our models more carefully, possibly using advanced tuning techniques.

Which Objects Were Hard to Recognize?: Our data showed that some objects, like cats and dogs, got mixed up more often. This hints at areas where we could focus our research and maybe develop special techniques for certain object types.

Concluding Reflections

This project taught us a lot, showing both what we know about deep learning and what we still have to discover. It reminded us how important it is to keep trying different approaches, test our models thoroughly, and stay updated with the latest research in the world of neural networks.