Project I | Deep Learning: Image Classification with CNN Project Report

Description of the chosen CNN architecture.

For this image classification project, the chosen CNN architecture was a modified VGG16 architecture, which leverages the power of transfer learning and fine-tuning. The model is initialized with pre-trained weights from the ImageNet dataset, providing a strong foundation for recognizing general image features. To adapt to the CIFAR-10 dataset, the last two convolutional blocks are fine-tuned while the initial layers remain frozen. The original fully connected layers are replaced with a streamlined architecture including Global Average Pooling, a Dense layer with regularization, Batch Normalization, and Dropout to reduce overfitting. This approach leverages the knowledge from a vast dataset while efficiently adapting to the specific characteristics of CIFAR-10, promoting both speed and accuracy in image classification.

Explanation of preprocessing steps.

To ensure optimal model performance, we implemented a series of preprocessing steps to enhance the quality and suitability of the CIFAR-10 dataset. We began by loading the dataset using `tensorflow.keras.datasets` and partitioning it into training and testing sets. Before any transformations, we visualized sample images using the `pyplot` module to gain a preliminary understanding of the data. Next, we applied image augmentation using the Keras library, introducing random rotations, shifts, flips, and zooms during training. This technique enhances the model's robustness to variations in the data, improves generalization to unseen images, and mitigates overfitting. Furthermore, we normalized the pixel values in both the training and testing datasets to a range between 0 and 1 as data normalization is crucial for optimizing the training process and improving the model's overall performance.

Details of the training process (e.g., learning rate, batch size, number of epochs).

The model training employs a learning rate of 0.00001 during the fine-tuning phase and a batch size of 64. The training process runs for a maximum of 75 epochs, but early stopping is used to prevent overfitting. In our case, training usually stopped at around epochs in the high 40s to low 50s due to the early stopping function from the Keras Callbacks module being implemented. The Adam optimizer is used to adjust model weights, minimizing the categorical cross-entropy loss function. To further improve model generalization, L2 regularization (with a factor of 0.0005) and dropout (with a rate of 0.4) are applied. Data augmentation techniques, including rotation, horizontal flips, width/height shifts, and zooming, are used to increase the diversity of the training data and enhance the model's robustness. These hyperparameters and techniques are carefully chosen to achieve optimal model performance on the CIFAR-10 image classification task.

Our final model, utilizing a fine-tuned VGG16 architecture, achieved an accuracy of 82.51% on the CIFAR-10 test set, with a corresponding loss of 0.5800. This performance represents a significant improvement over our initial explorations:

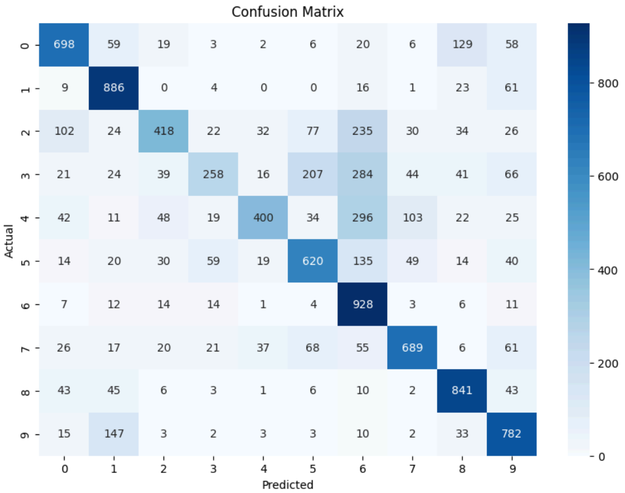
- **Baseline CNN:** A custom-built convolutional neural network yielded an accuracy of approximately 62%, highlighting the limitations of a basic approach for this dataset.

- **VGG16 (Feature Extraction):** Leveraging the pre-trained VGG16 model for feature extraction resulted in a notable improvement, reaching around 72% accuracy. However, this approach still fell short of our desired performance.
- **VGG16 (Fine-tuning):** Fine-tuning the last two convolutional blocks of the VGG16 architecture proved to be the most effective strategy, ultimately achieving an accuracy of around 82.51%.

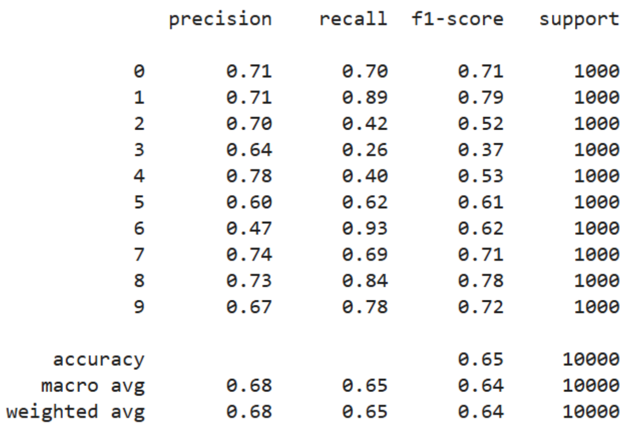
Evaluation and Future Directions

Further analysis using a confusion matrix revealed an overall accuracy of 83% for the final model using the VGG16 pre-trained model, indicating reasonably good generalization capabilities on the CIFAR-10 dataset. While the model excelled in classifying categories like airplane, automobile, ship, and truck (precision and recall exceeding 0.85), it faced challenges distinguishing between visually similar classes such as cat vs. dog and bird vs. airplane. These observations suggest potential areas for future improvement. In the future, we could explore advanced data augmentation techniques, fine-tune hyperparameters further, or consider architectural modifications to address class-specific challenges and elevate the model's overall performance.

Baseline CNN Confusion Matrix

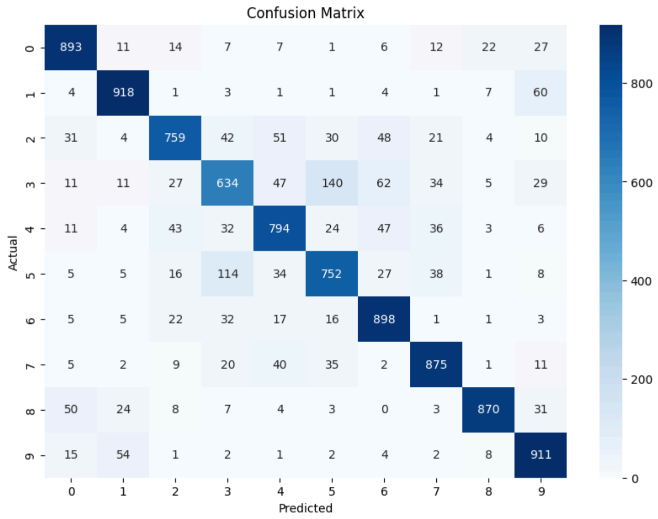


	0	1	2	3	4	5	6	7	8	9
0	698	59	19	3	2	6	20	6	129	58
1	9	886	0	4	0	0	16	1	23	61
2	102	24	418	22	32	77	235	30	34	26
3	21	24	39	258	16	207	284	44	41	66
4	42	11	48	19	400	34	296	103	22	25
5	14	20	30	59	19	620	135	49	14	40
6	7	12	14	14	1	4	928	3	6	11
7	26	17	20	21	37	68	55	689	6	61
8	43	45	6	3	1	6	10	2	841	43
9	15	147	3	2	3	3	10	2	33	782

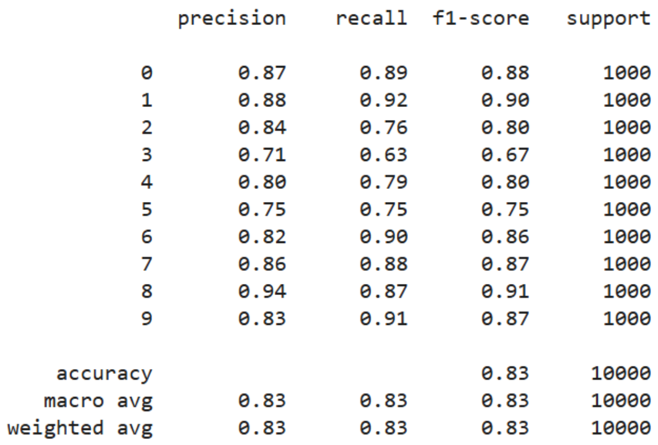


```
              precision    recall  f1-score   support

           0       0.71      0.70      0.71      1000
           1       0.71      0.89      0.79      1000
           2       0.70      0.42      0.52      1000
           3       0.64      0.26      0.37      1000
           4       0.78      0.40      0.53      1000
           5       0.60      0.62      0.61      1000
           6       0.47      0.93      0.62      1000
           7       0.74      0.69      0.71      1000
           8       0.73      0.84      0.78      1000
           9       0.67      0.78      0.72      1000

    accuracy                           0.65     10000
   macro avg       0.68      0.65      0.64     10000
weighted avg       0.68      0.65      0.64     10000
```

VGG16 Fine Tuned Model Confusion Matrix



Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	893	11	14	7	7	1	6	12	22	27
1	4	918	1	3	1	1	4	1	7	60
2	31	4	759	42	51	30	48	21	4	10
3	11	11	27	634	47	140	62	34	5	29
4	11	4	43	32	794	24	47	36	3	6
5	5	5	16	114	34	752	27	38	1	8
6	5	5	22	32	17	16	898	1	1	3
7	5	2	9	20	40	35	2	875	1	11
8	50	24	8	7	4	3	0	3	870	31
9	15	54	1	2	1	2	4	2	8	911



```
              precision    recall  f1-score   support

           0       0.87      0.89      0.88      1000
           1       0.88      0.92      0.90      1000
           2       0.84      0.76      0.80      1000
           3       0.71      0.63      0.67      1000
           4       0.80      0.79      0.80      1000
           5       0.75      0.75      0.75      1000
           6       0.82      0.90      0.86      1000
           7       0.86      0.88      0.87      1000
           8       0.94      0.87      0.91      1000
           9       0.83      0.91      0.87      1000

    accuracy                           0.83     10000
   macro avg       0.83      0.83      0.83     10000
weighted avg       0.83      0.83      0.83     10000
```

What is your best model. Why?

Our experiments led us to select a fine-tuned VGG16 architecture as the most effective model for CIFAR-10 classification. By fine-tuning the last two convolutional blocks, we achieved a remarkable test accuracy of 82.51%, significantly surpassing a standard CNN and VGG16 with feature extraction alone. This superior performance stemmed from enhanced generalization, enabled by the adaptation of pre-trained features to the CIFAR-10 dataset. Regularization with L2 and dropout, coupled with data augmentation, further promoted robustness and prevented overfitting. Moreover, our efficient training strategy, incorporating early stopping and an adaptive learning rate, optimized convergence and resource utilization. Ultimately, this combination of factors resulted in a highly accurate and robust model, ideally suited for our CIFAR-10 classification task.

Insights gained from the experimentation process.

During the experimentation process, we must highlight the significant advantages of transfer learning and fine-tuning for image classification. While a custom CNN model struggled to achieve high accuracy, leveraging a pre-trained VGG16 architecture as a base significantly improved performance. Moreover, fine-tuning the last two convolutional blocks of VGG16 proved more effective than feature extraction alone, leading to a substantial accuracy boost. Augmenting the training data with small transformations like rotation, zoom, and flips enhanced the model's ability to generalize to unseen data. Finally, careful hyperparameter tuning, including reducing L2 regularization and the learning rate, prevented over-penalization and further optimized accuracy.