ML Model Optimization

Problem Outline

This project focuses on leveraging the capabilities of CNNs to analyze and categorize images from the CIFAR-100 dataset, a challenging and diverse collection of 100 different object classes. The CIFAR-100 dataset, known for its complexity and variation, presents a unique opportunity to explore the depths of machine learning algorithms in image classification.

In this project I was provided with a base CNN model, and was tasked to improve it's performance on the CIFAR 100 dataset, this took form of the following tasks:

1. Training the base model to get the baseline performance, this will serve as the target to beat.
2. Comparing the performance of my improved model against the baseline model at 15 epochs.
3. Achieving a test set accuracy higher than 44% average, while displaying the best class and worst class accuracy.
4. Determining the performance of a snapshot ensemble of 5 models.

Approach

Deep learning, especially in the field of computer vision has had many advances that have constantly pushed the performances of models in the domain of image classification. My approach to improve the performance of the baseline model was to systematically incorporate modern deep learning architectural practices into the machine learning model while documenting the effects of each change.

Experiment Details

In this experiment, we utilize the CIFAR-100 dataset, renowned for its diversity and complexity, comprising 100 distinct classes of images. The dataset is ingeniously partitioned to optimize the training and validation process. Specifically, it is divided as follows: the training set is a combination of the first 35% and the last 55% of the original training data, ensuring a broad and varied range of images for the model to learn from. For validation, a segment constituting 10% of the training data (ranging from 35% to 45%) is employed, providing a reliable subset for performance evaluation during the training phase. The test set remains the standard set of 10,000 images, serving as an unbiased metric for the final assessment of the model's classification prowess.

All models that were trained using the categorical cross-entropy loss function, while utilizing the Adam optimizer with a base learning rate of 0.001. As seen in the experiments, this proved too high cause the training to progress in an unstable manner, this was later reduced to 0.0004 for all experiments. Furthermore, the models were initially trained using 15 epochs, however, this later proved to not be enough to allow the model to reach its full potential, this this was later increased. Lastly, model checkpointing was used to save the best performing model (given by validation loss), and this was the model that was used on the test set.

Base Line Model

The baseline model was a weak classifier consisting of convolutional layers and dense layers. It starts with an input layer shaped to accommodate the dataset images, followed by two core convolutional layers with 32 and 64 filters, each paired with batch normalization and ReLU activation. Max pooling layers reduce the data dimensions, aiding in computational efficiency. The network then flattens the data for a dense layer with 512 neurons, incorporating batch normalization, ReLU activation, and a dropout rate of 50% to avoid overfitting. The final layer, designed for classification, is a dense layer with 100 neurons using softmax activation. Overall, this model only two convolutional layers followed by a Dense layer, this model doesn't have a strong feature extraction capability and is prone to overfitting as will be shown by its performance on the test and validation sets.

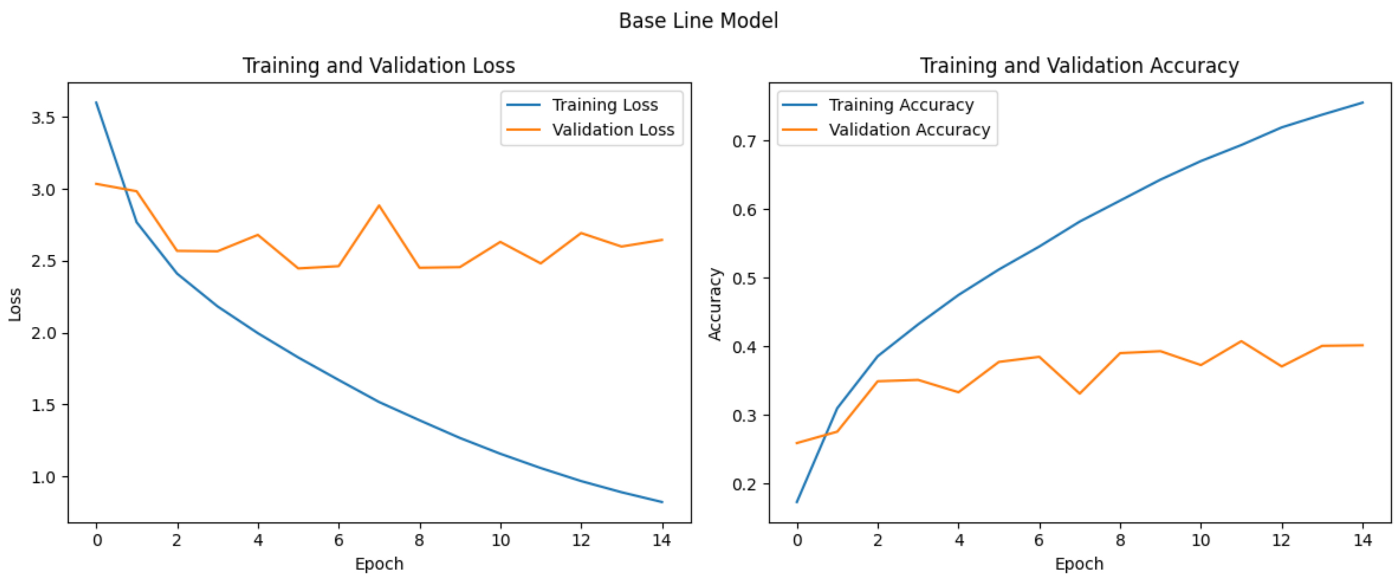
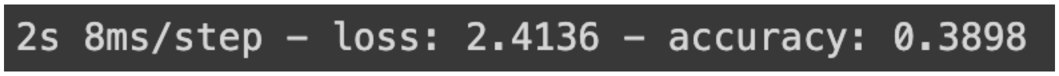


Figure 1: Base Line Model Training and Validation Curves

The training progression curves display the progression of the training and validation loss and accuracy throughout the training of the model. As can be seen in the graph, the model overfits rather quickly around the second epoch, this is when both the validation loss and validation accuracy stop improving while the training metrics continually improve, a classic sign of overfitting. Additionally, the oscillation of the validation suggests that training is a bit unstable at this learning rate. Due to this reason, we reduce the learning rate to 0.0004 as well as adding a decay such that the learning rate is decreased by a factor 1e-6 throughout the training progression and rerun the experiment.

Figure 2 still demonstrates overfitting, however, the instability of the training has decreased as can be seen by the lesser extent of the oscillation and the overfitting occurs 2 epochs later. As for the performance on the test set, the baseline model achieves an accuracy of 39% which is slightly lower than the specified target, adding more epochs to this model would also prove futile as shown by the training curves.



```
2s 8ms/step - loss: 2.4136 - accuracy: 0.3898
```

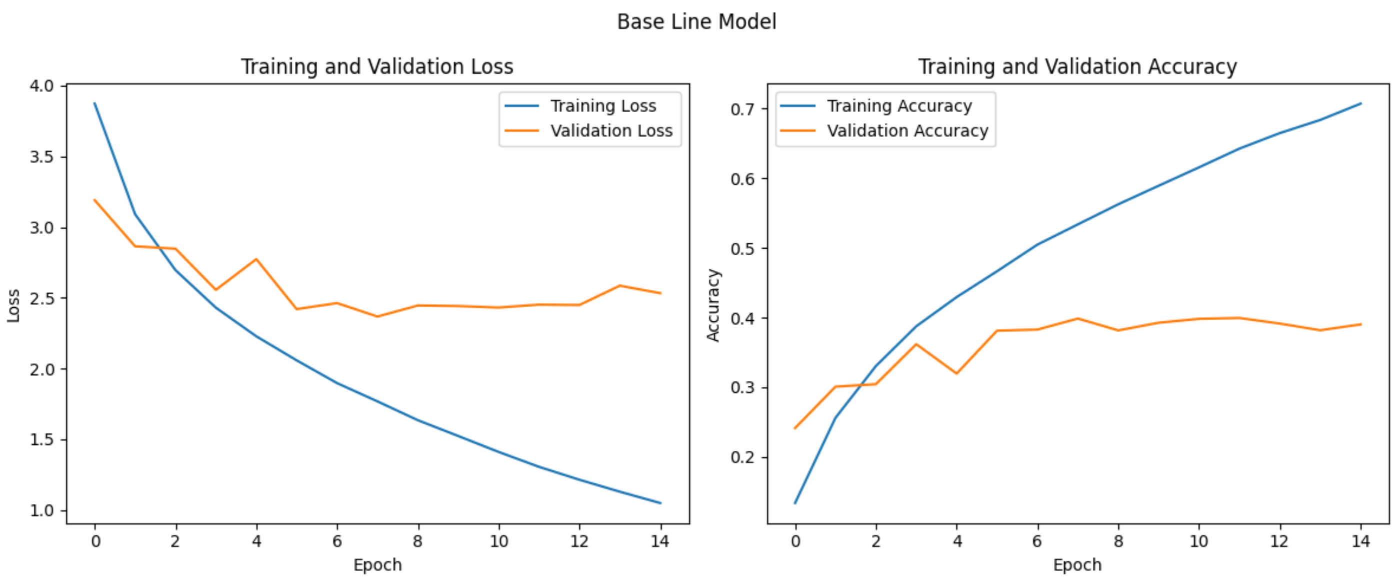
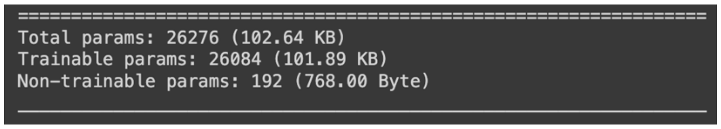


Figure 2: Base Line Model Training and Validation Curves with Smaller Learning Rate

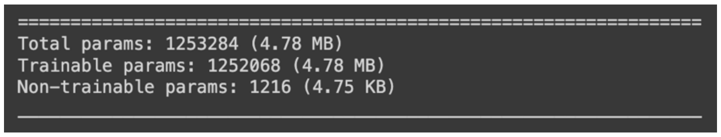
Model Improvements

In this section I outline the different modifications gradually made by the model as well as brief performance metrics, this section will not provide extensive evidence such as training progression curves for the sake of brevity.

1. The very first change added to the model was swapping out the Dense and Flatten layers with a single global average pooling layer. The global average pooling computes the mean value of each feature map. This not only conserves computational resources and memory but also aids in mitigating overfitting, thereby fortifying the model's generalization capabilities. This led to the prevention of overfitting, however, it outlined the weak feature extraction capabilities of the base model as this one only achieved 20% accuracy on the test set. Lastly it also resulted in a massive decrease in model parameters.



```
=================================================================
Total params: 26276 (102.64 KB)
Trainable params: 26084 (101.89 KB)
Non-trainable params: 192 (768.00 Byte)
_________________________________________________________________
```



```
=================================================================
Total params: 1253284 (4.78 MB)
Trainable params: 1252068 (4.78 MB)
Non-trainable params: 1216 (4.75 KB)
_________________________________________________________________
```

2. The second change to the model was stacking to convolutional units in each layer in an attempt to increase the feature extraction capabilities. When trained for 15 epochs, this model did not overfit and showed that 15 were not enough for it to reach its maximum potential, therefore, the epochs were increased to 50, following training for 50 epochs, this model was able to improve upon the baseline model achieving a test performance of 41%

3. The third change involved adding one more convolutional layer to make it a total of 3 layers once again to increase the feature extraction capabilities, after 50 epochs, the model with this change achieved a 50% testset accuracy.
4. As the experiments were heading in a direction in which the model was getting deeper, the next change was to incorporate residual connections to further mitigate overfitting and allow the model to prevent the vanishing gradient problem. This model achieved a test-set accuracy of 52%, however, it was once again showing signs of overfitting.
5. To mitigate overfitting and increase the generalization ability of the model the next modification was the addition of data augmentation this included
    a. Random Flipping horizontally
    b. Random Rotation with 0.2 radians
    c. Random Zooms by 10%
    d. This model mitigated the overfitting however, it still showed no sign of reaching its maximum potential at the end of 50 epochs, thus I trained it for a further 100 epochs, the final model achieved a 57% accuracy.
6. I next experimented with incorporating depthwise-seperable convolutions however, this didn't result in an improvement, and decreased model performance on the test set to 49%.
7. The final change was increasing the number of convolutional layers to 5 layers, adding 25% dropout to the final 2 convolution layers, 20% and 15% dropout to the third and second layers respectively, as well as swapping the activation function in the convolution layers from ReLU to ELU (exponential linear unit), this was done to mitigate the dying ReLU problem. Trained on 50 epochs, this model achieved a 63% test-set accuracy, with signs of greater accuracy given more epochs.

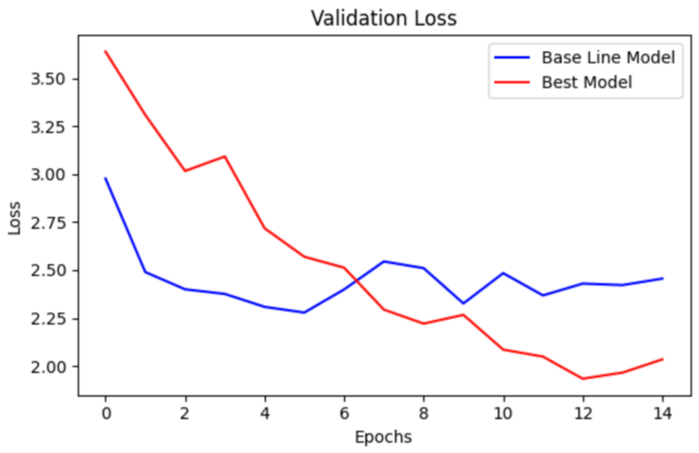
```
200/200 [==============================] - 3s 11ms/step - loss: 1.3594 - accuracy: 0.6259
```

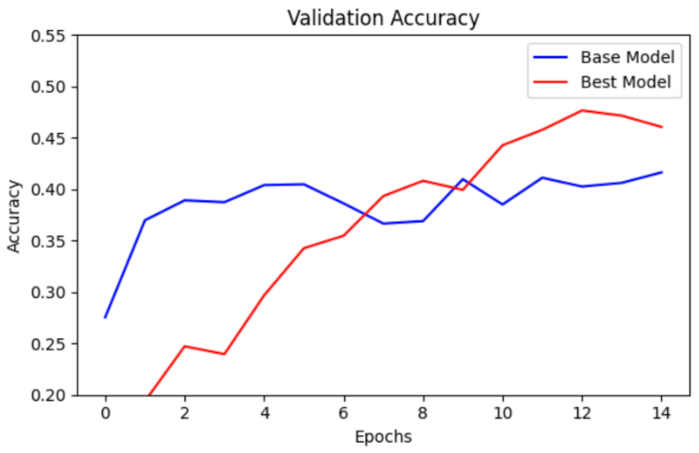
Modification	**Test-Set %**	**Model Parameters**	**Notes**
Global Average Pooling	**20%**	**30000**	**Mitigated Overfitting**

Comparison At 15 Epochs

In this section we compare the performance of the base model and my best model at 15 epochs.

Model	Test-Set Performance
Base Model	39%
Best (Deep Residual) Model	51%





As can be seen my best trained model achieves both a lower validation loss, as well as a higher validation accuracy compared to the baseline model.

```
3s 10ms/step - loss: 1.7804 - accuracy: 0.5101
```

Best and Worst Class Accuracies

The best class accuracy achieved by the best model is 93%, this is for the 68th class which is for the physical object of “pine_trees”, and the worst class accuracy is 24% for the 72nd class which corresponds to the “maple_tree”.