Introduction

For this assignment, my task was to classify images based on the CIFAR-100 dataset. This dataset consists of 60,000 32x32 color images divided into 100 classes, with each class containing 600 images. The dataset is split into 50,000 training images and 10,000 testing images, with each class containing 500 training and 100 testing images. My goal was to design, implement, and evaluate a Convolutional Neural Network (CNN) to classify the images effectively. While researching CNN architectures, I was intrigued by the VGG architecture. However, due to its high computational cost and long training times, I opted for a simpler architecture.

Method

Given the computational constraints, I designed a CNN with a simpler architecture comprising 8 convolutional layers. I incremented the number of filters (32, 64, 128, 256) after every two convolutional layers. Each convolutional layer is followed by a BatchNormalization layer to mitigate overfitting and a ReLU activation layer to introduce non-linearity. MaxPooling layers are added after every two convolutional layers to reduce the spatial dimensions and emphasize the strong features of the images. Dropout layers are included after each MaxPooling layer to further reduce overfitting. After these convolutional layers, a fully connected layer is added, followed by batch normalization, activation, and dropout layers. The final layer is a dense output layer with softmax activation for classification.

Experiment

To prepare the data for training, I loaded the CIFAR-100 dataset using Keras and performed one-hot encoding on the labels. To address the initial overfitting observed, I employed data augmentation techniques, such as random rotations and vertical/horizontal shifts, to artificially expand the training dataset. The model was compiled using the Adam optimizer with a learning rate of 0.001. Training was conducted for 100 epochs with a batch size of 64. Early stopping was enforced if there was no improvement in validation loss for 10 consecutive epochs, and the best model weights (from the epoch with the lowest validation loss) were restored. Additionally, the learning rate was reduced by 20% if accuracy did not improve after 5 consecutive epochs, with a minimum learning rate set to 0.0001.

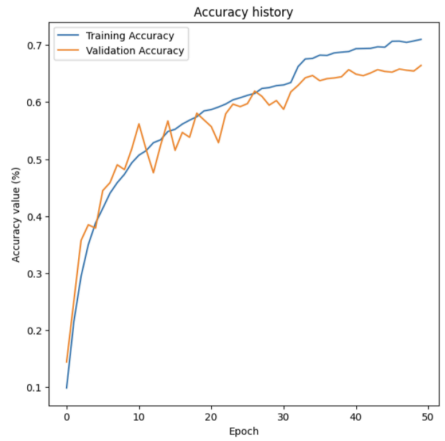
Results

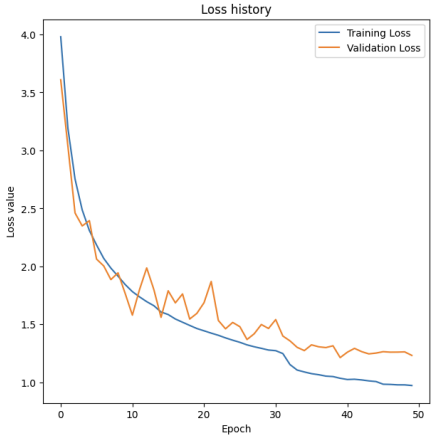
After training the model multiple times, the best results were achieved with an early stoppage at 50 epochs, yielding a test loss of 1.213 and a test accuracy of 65.6%. The best training accuracy/loss was 80% and 0.65 respectively. There could be a case that there is a slight overfitting of the model, but this was the best balance I could achieve between overfitting/underfitting the model.

Conclusion

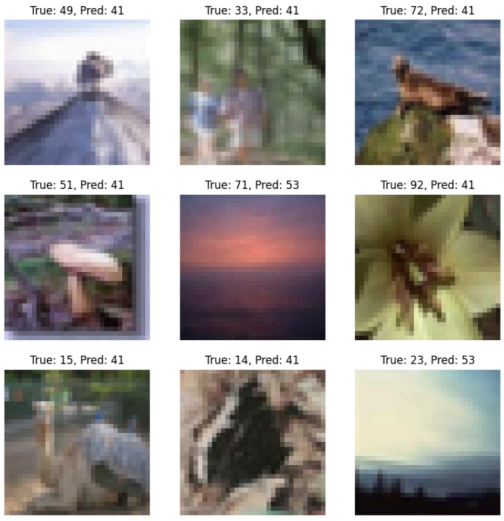
Through many trial and error runs, I found that this model perfectly balanced the overfitting issues and overall accuracy of the model as with my previous two models, I encountered both spectrums. My first model had extremely high training accuracy but a very low testing accuracy(90% and 50% respectively). This was due to the fact that I made my model extremely complex and didn't have many techniques of reducing overfitting. I then added data augmentation techniques, batch normalization layers, and drop out layers to solve this. I also made the model a bit less complex with less layers.My second model, while not overfitted, wasn't as accurate as the current one as it yielded 52% training and 48% testing accuracy. This result was what I wanted as I fixed the overfitting problem, but was left with bad performance on the model. I realized I overcompensated on the overfitting techniques so I reduced the dropout rates and added a few more layers to the model. I also tweaked the learning rate a bit and went with the Adam optimizer instead of the RMSprop optimizer. This led me to my current results. While the model successfully classified CIFAR-100 images without significant overfitting or underfitting, the overall accuracy indicates the potential for further enhancements. Future improvements could involve experimenting with different batch sizes, learning rates, and dropout rates. Additionally, exploring more complex model architectures with deeper layers could further improve performance.

```
Training loss: 0.6589261889457703
Training accuracy: 0.8027200102806091
Test loss: 1.2133790254592896
Test accuracy: 0.6568999886512756
```





Incorrectly Classified Examples



Correctly Classified Examples