Machine Learning Final Project Report

Introduction

The main objective of this project is to improve the performance of the Convolutional Neural Network (CNN) model for CIFAR-10 dataset classification. The previously built CNN model as part of the final assignment for the machine learning course obtained accuracy of 75%. I experimented with various approaches, including but not limited to updating CNN architecture, implementing dropout layers, max-pooling, etc. With the finalized architecture, the model's performance improved to around **91% accuracy,** which is a significant improvement from the **previous accuracy of 75%**.

Previous Image Classification Model

The CNN model built for the Final Homework consisted of six convolutional layers, two fully connected layers, and two max-pooling layers. The input was fed to four convolutional layers, followed by a max-pooling layer, a fifth convolutional layer, a second max-pooling layer, a sixth convolutional layer, and the final two fully connected layers.

As for the hyperparameters, learning rate=0.01, momentum = 0.9, weight decay=5e-4, 50 epochs, 128 batch size, and Stochastic Gradient Descent (SGD) as the optimizer were used.

ReLU was used as an activation function for all the convolution layers and the first fully connected layer. Softmax was used as the activation function for output from the final fully connected layer.

The given model achieved an **accuracy of 75%**.

Approaches Experimented

I tried several approaches to improve the accuracy of the CNN model provided in Final Homework. The first approach was to use different activation functions and gradient optimization techniques. I experimented with Adam and RMSprop algorithms. However, with Adam, the accuracy didn't improve from 64% whereas RMSprop reached accuracy of 83%. Surprisingly, Stochastic Gradient Descent (SGD) seemed to be the best optimizer for this use case with CIFAR-10 dataset. I also compared Tanh and ReLU activation functions and found ReLU slightly more efficient.

Experimentation with different learning rates and weight decay were also carried out with learning rates of 0.01, 0.005, and 0.001 and weight decay of 5e-3, 5e-4, and 5e-5. Learning rate of 0.005 and weight decay of 5e-4 were found to have best results.

Different CNN architectures, such as changing the number of convolution layers along with their kernel size and padding and adding fully connected layers were also experimented. I implemented residual neural networks, which significantly improved model performance and helped tackle degradation in deep neural networks. I tried adding one residual block followed by two residual blocks. Two residual blocks had the best performance overall, with an accuracy of around 78%.

Then, tests using batch normalization after each convolutional layer and max-pooling to downsample the feature map were performed. I also experimented with different dropout layers with different rates to prevent overfitting. The model performance with different techniques is detailed in Table 2. The final architecture found to have the best overall accuracy is explained in detail in the next section.

Final Model Architecture

The final CNN architecture for the CIFAR-10 dataset has eight convolutional layers (including two residual blocks) and two fully connected layers. Data augmentation techniques using RandomCrop and RandomHorizontalFlip are implemented, which helps to increase the training dataset by adding slightly modified copies of the original dataset. It is a great technique to improve data diversity and reduce overfitting.

The output of each convolutional layer is followed by batch normalization, which helps to make the training process faster and more stable. ReLU is the activation function for all the convolutional layers and the residual blocks. After each pair of convolutional layers (2 convolutional blocks in succession), max-pooling is implemented.

After the first two convolution layers, a residual block made up of two convolutional blocks with kernel size 3 is added. The residual block helps to solve the exploding or vanishing gradient and hence helps avoid overfitting issues. This is followed by the second pair of convolutional layers, followed by a second residual block with kernel size 5. Each of the residual blocks is followed with a dropout of 10%.

The last two layers are fully connected. The first fully connected layer is followed by a dropout of 10% and uses ReLU as an activation function. The final layer uses log softmax for activation.

Configurations:

Epochs: 50
Batch Size: 128
Learning rate: 0.005
Weight decay: 5e-4
Dropout rate: 0.1
Activation function: ReLU (Log Softmax in the output layer)
Optimizer: Stochastic Gradient Descent (SGD) with 0.9 momentum

The detailed design of the model is shown in Figure 1.

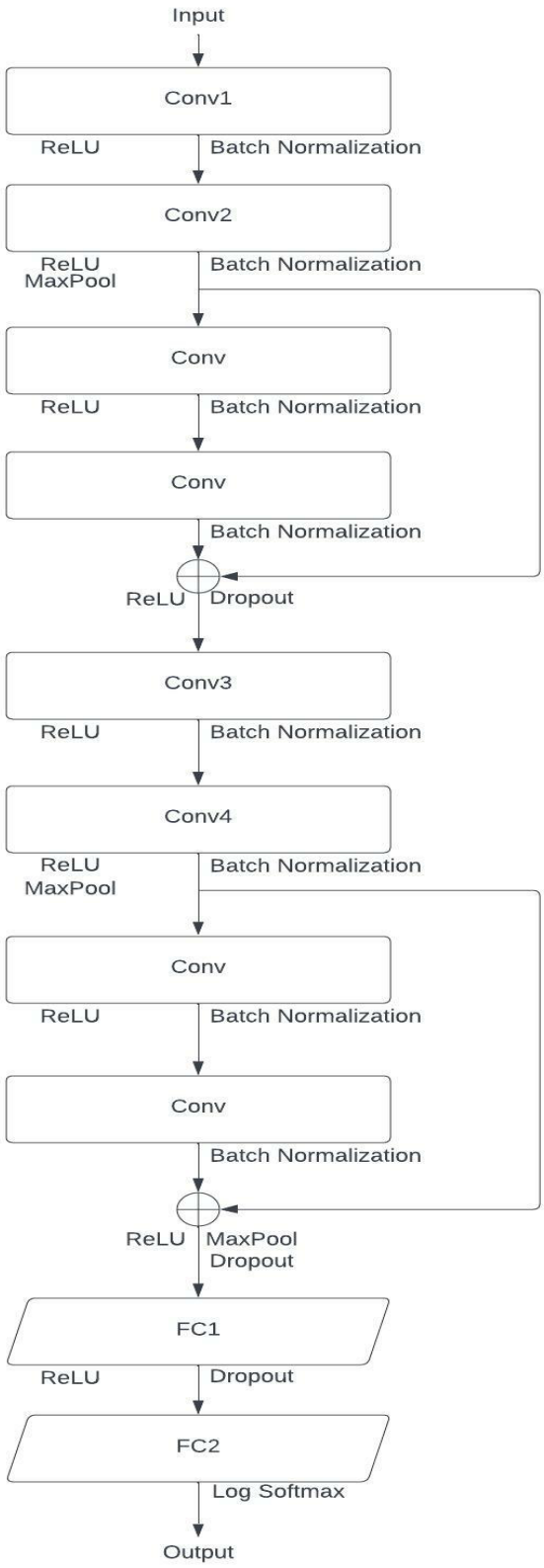

Figure 1 Model Architecture

Layer		Configuration	Size of Output data	Size of parameters
Conv1		Number of Filters: 64 Kernel Size: 3 Stride: 1 Padding: 1	32*32*64	1792
Conv2		Number of Filters: 128 Kernel Size: 3 Stride: 1 Padding: 1	32*32*128	73856
Maxpool1		Kernel Size: 2 Stride: 2 Padding: 0	16*16*128	0
ResBlock1	Conv	Number of Filters: 128 Kernel Size: 3 Stride: 1 Padding: 1	16*16*128	147584
	Conv	Number of Filters: 128 Kernel Size: 3 Stride: 1 Padding: 1	16*16*128	147584
Conv3		Number of Filters: 256 Kernel Size: 5 Stride: 1 Padding: 1	14*14*256	819456
Conv4		Number of Filters: 512 Kernel Size: 5 Stride: 1 Padding: 1	12*12*512	3277312
Maxpool2		Kernel Size: 2 Stride: 2 Padding: 0	6*6*512	0
ResBlock2	Conv	Number of Filters: 512 Kernel Size: 3 Stride: 1 Padding: 1	6*6*512	2359808
	Conv	Number of Filters: 512 Kernel Size: 3 Stride: 1 Padding: 1	6*6*512	2359808
Maxpool3		Kernel Size: 4 Stride: 4 Padding: 0	1*1*512	0
FC1		Input Features: 512	512	262656

	Output Features: 512 Neurons: 512		
FC2	Input Features: 512 Output Features: 10 Neurons: 10	10	5130

Table 1 Model Configuration and output sizes

Results

The above CNN architecture with convolutional layers, residual block and implementation of batch normalization, dropout and maxpooling resulted in the **accuracy of 91.02%** for the given CIFAR-10 dataset. The loss curve for the final model is presented in Figure 2.

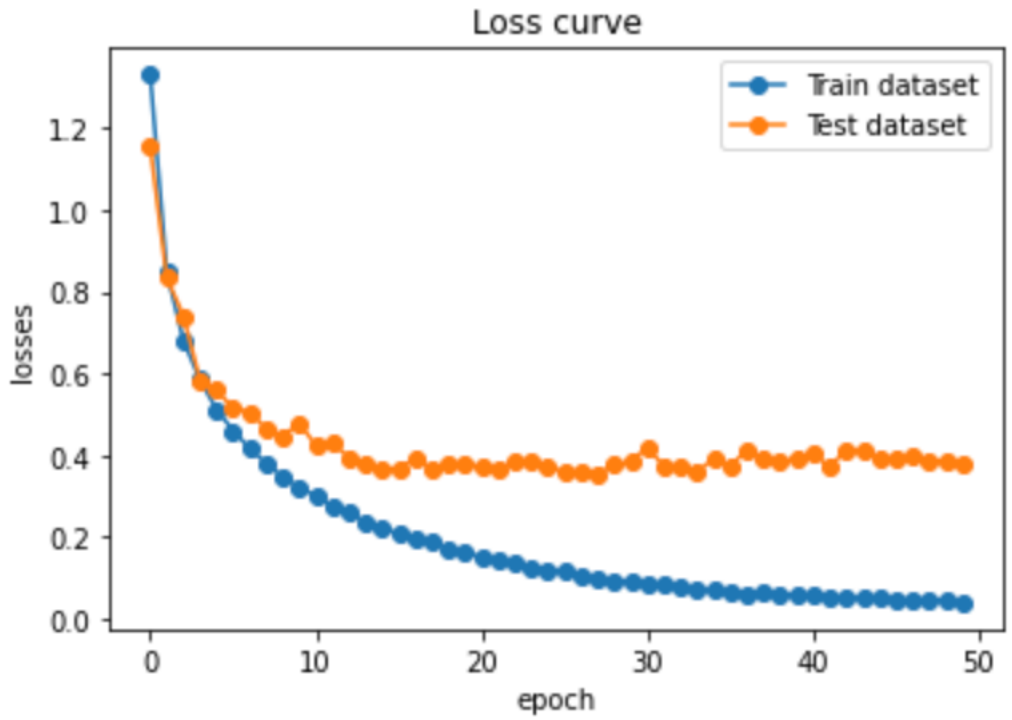

Figure 2 Loss curve for final neural network architecture

Based on the results, residual blocks helped reduce overfitting problems whereas batch normalization provided better performance, helped reach training convergence faster and allowed wider range of learning rates to be used. The addition of dropout layer contributed to make the model more robust and less prone to overfitting.

The comparison of different experiments carried out; primarily to determine the impact of the implementation of dropout layers, and batch normalization are shown in Table 2. All the tests are carried out with the same hyperparameters used for the final CNN architecture. Each model has the same configuration for convolutional layers and residual blocks.

Model Configuration	Time to Train (min)	Accuracy (%)
Without batch normalization and dropout	33.41	78.41
With batch normalization and without dropout	35.3	89.44
With dropout and batch normalization	35.65	91.02

Table 2 Comparison with different model configurations

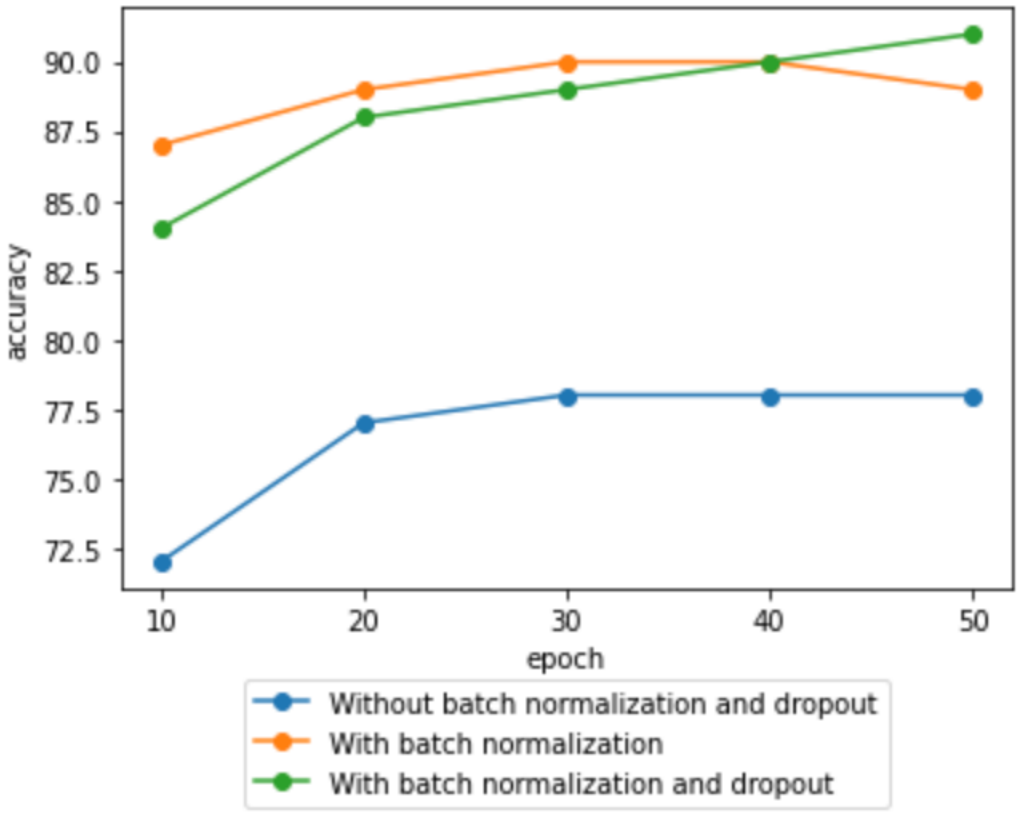

Figure 3 Comparison of accuracy for different model configurations

Conclusion

I was able to build a deep neural network that significantly outperformed the CNN architecture built in Final Homework. The new CNN architecture is different from the Final Homework and is based on the Residual Network (ResNet) architecture. Different optimization techniques, such as data augmentation, batch normalization, max-pooling, and dropout are also applied, which improved the model's performance. The new model improved the accuracy of image classification with the CIFAR-10 dataset around 91%. Hence, the selection of CNN architecture, along with the proper approaches based on the machine learning task at hand and the available dataset, can have a considerable impact on the performance of the model.