**Computer Vision Final Project Report**

**Mengjie Shen**

**Project Outline:**

1. **Setting up and Loading data**
2. **Lenet Structure with SGD as optimizer**
3. **Lenet Structure with ADAM as optimizer**
4. **Modified Lenet structure**
5. **VGGnet-19 Structure**
6. **Conclusion**
7. **Setting up and Loading data**



I divide the train dataset to train and validation set, with size of 45000 and 5000 each. Validation set is used to determine the hyper parameter and when to stop training.



1. **Lenet Structure with SGD as optimizer**

I first try to implement the Lenet structure. This is my code implementation down below:

图片包含 文本

描述已自动生成

I set the learning rate to be 0.0003. I stop the training at the epochs around 50 because I observe that the validation loss started to increase at the epochs of around 50.

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图片包含 表格

描述已自动生成

图形用户界面, 应用程序, Teams

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图片包含 形状

描述已自动生成

1. **Lenet Structure with ADAM as optimizer**

Next, I try to improve my Lenet structure by changing SGD optimizer to Adam optimizer. Since Adam can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum, it can speed up my gradient descent and decrease my learning rate after I reach certain epochs.

I also found that validation loss can’t directly and clearly display where is the best stopping point, I used validation accuracy instead of validation loss to determine when I should stop the model to avoid overfitting and underfitting.

**图形用户界面, 文本, 应用程序

描述已自动生成**

**文本

描述已自动生成**

**图片包含 游戏机

描述已自动生成**

I adopted the early stopping after I observe that validation accuracy no longer increases. However, the overall accuracy didn’t improve.

**图表, 折线图

描述已自动生成图表

描述已自动生成文本

描述已自动生成**

1. **Modified Lenet structure:**

I changed the structure of the original model a little bit. I change the average pooling layer to max pooling, change tanh activation function to relu activation function. I also increase the output channel for each layer. I added a dropout layer in order to avoid the issue of overfitting.

**图形用户界面

低可信度描述已自动生成**

I tuned the learning rate.

1. Trained with lr: 0.001

**表格

描述已自动生成**

Accuracy:

**文本

描述已自动生成**

Train loss curve and validation accuracy curve:

**图表

描述已自动生成** **图表, 折线图

描述已自动生成**

1. Train with lr: 0.0005:

**表格

描述已自动生成**

Accuracy:

**文本

描述已自动生成**

Train loss curve and validation accuracy curve:

**图表

描述已自动生成** **图表

描述已自动生成**

1. Trained with lr: 0.0001

图片包含 窗户, 大, 游戏机, 站

描述已自动生成

一些文字和图案

描述已自动生成

Accuracy:

**文本

描述已自动生成**

Train loss curve and validation accuracy curve:

**图表

描述已自动生成** **图表

描述已自动生成**

Conclusion: lr with 0.0001 has the best performance.

1. **VGGnet-19 Structure**

Next, I implemented VGGnet-19 for image classification. The code down below is my structure for the VGGnet. Since there are more layers in VGGnet that Lenet, I used a for loop and passed into the parameter configuration to build the net (\_make\_layers function)



VGG-19 model training:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

The training process is relatively slow with GPU by google colab, so I only trained for 23 epoches after I found that the validation accuracy no longer has much of increases.

文本

描述已自动生成

Result for VGGnet-19:

We can observe that accuracy greatly improved.

图片包含 文本

描述已自动生成

Non-negative log likelihood loss for training dataset and accuracy for validation dataset:

图片包含 图表

描述已自动生成图表

描述已自动生成

1. **Conclusion:**

VGG-19 model has achieved the best accuracy of 81.6%. This model could potentially achieve the higher result because I didn’t train this model to its maximum due to the limited use of GPU by Google Colab. The modified Lenet with learning rate of 0.0001 also achieved a solid result of 70.96%. Future improvement will be adding batch normalization to the model to see whether it can achieve a greater improvement.