Feature design: preprocessing

Algorithm:

when it comes to image recognition, a convolutional neural network is one of the best options. For this project, a 10-convolutional layer CNN was built. There are 5 max pooling layers in total, one after every two convolutional layers to reduce the spatial size, each pooling layer will exactly reduce the input length and width by half. Also, there are 5 dropout regulations being applied, one after every two convolutional layers which has a drop out probability of 0.25, and one dropout after the fully connected layer which has a drop out probability of 0.5, the team believe this will prevent overfitting and reduce the interdependent learning amongst the neurons [1]. As shown in the flow chart 1 below, there are max pooling layers after convolutional layer 2, 4, 6, 8, 10, and dropout regulations after convolutional layer 3,5,7,9 and the fully connected layer. Activation function rectified linear unit (ReLU) is used after every convolutional layer and the fully connected layer, and batch normalization is applied after the activation function.

Methodology:

Firstly, custom training set and validation set was produced, the training set has 95% of the data and the validation set has 5%, the train set is set to be 95% of the data because the CNN network have a huge amount of parameters and the team wants to use as many data as possible to train the model. Moreover, 5% of the original training dataset is still 2500 data points, it can produce a reasonable result in a short amount of time.

Data is fed into the CNN network a batch at a time, using the pytorch data loader class, and the batch size is set to be 300.

Hyperparameters:

As shown in the tables below there are close to 100 hyperparameters, it is impossible to tune them all. The first filter size is chosen to be 5 because according to a study from Cornell University [3], on mnist dataset 5x5 filter size gives the best result as shown in chart 1. And the study also showed that there is a trade of between complexity and accuracy, but if the model gets too complex without sufficient data points it has the risk of overfitting. The number of convolution layers was chosen by gradually increasing the number of convolution layers and record the performance on the validation set. The team has find 10 convolutional layers yields the best result, as shown in the chart 2, the performance increases until it reaches 12 layers, it overfits. While pooling layer was used to decreasing the number of neurons towards the bottom as suggested by Stefan Latter on research gate [2], to be specific, after every two convolution layers a pooling layer will be employed to reduce the number of neurons by half (decrease both length and width by half and increase the depth by 2). For the number of filters of the first convolution layer (the number for the rest of the convolution will just double every two layers), different values were tested, the accuracy was improving when the number of filters go from 8 to 64 for the first convolution layer, and the improvement was not very significant after the number passed 64, and the computing time increases exponentially as the number of the filters increases. Drop out is proven to be an efficient and effective strategy to prevent overfitting [4], thus it is used as the regulation strategy for this project, 25% of tensors get dropped every 2 convolution layers. Adam optimizer were used because it has an adaptive learning rate and momentum, it is shown to be more effective that other optimizers [5].

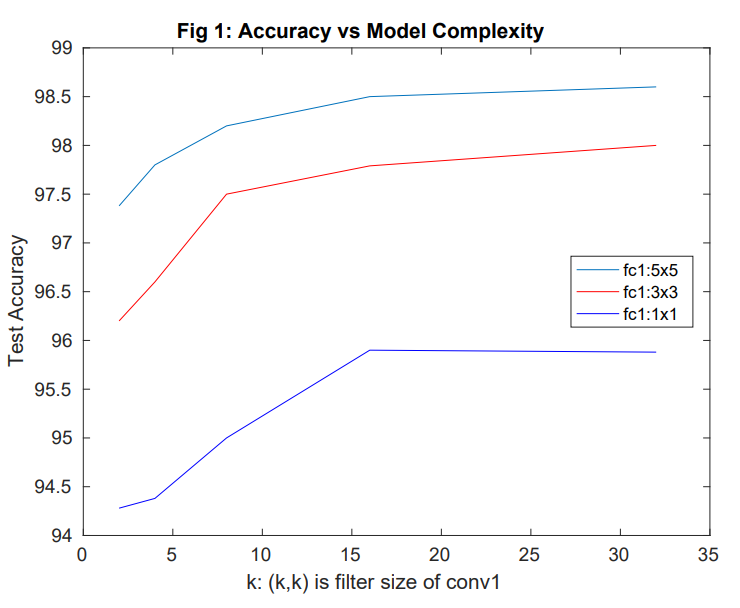


Chart 1 [3]

Discussion

Using the 10-layer CNN model have both advantages and disadvantages. The CNN network can be very accurate on image recognition tasks, our model is able to achieve an accuracy of 95.9% training on only 47500 samples, and there is still room for improvement if more time is given. The drawback of the model is the complexity, it is slow to train and have large memory footprint, according the paper from Atul Dhingra from Cornell University, there is a trade off between complexity and accuracy. Since there is no requirement about the speed or complexity in the assignment handout, a complex model with 10 layers was chosen.

Future work

If more time was given, the team will first focus on the augmenting the dataset, by creating rotated and translated versions of the original image. The cross entropy loss of the model was close to 0.001 during training, it has shown the signs of over fitting, if more data is produced by turning and shifting the original data, at the same time adding more regulations(dropout), then overfitting can be avoided.

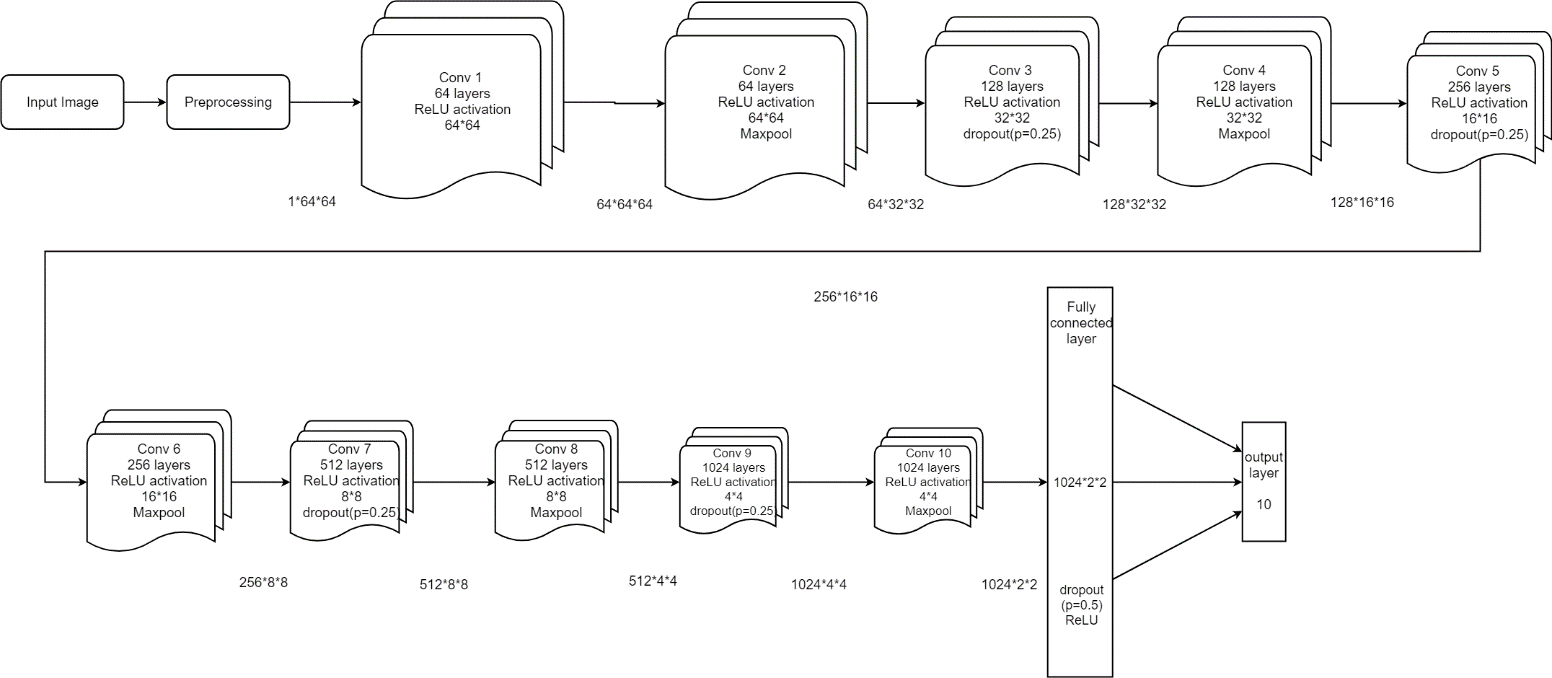
Chart 2

Results: More than ten models were trained and compared.

Confusion matrix will need to regenerate the dataset and retrain the model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***model*** | **layers** | **regulation** | **max pooling layer** | **accuracy** | **preprocessing** |
| *1* | 2 | n/a | 1 | 18.9 | n/a |
| *2* | 3 | n/a | 1 | 23.5 | n/a |
| *3* | 4 | 3 dropouts | 2 | 60.1 | n/a |
| *4* | 6 | 3 dropouts | 3 | 84.8 | n/a |
| *5* | 8 | 4 dropouts | 4 | 93.4 | threshold |
| *6* | 10 | 5 dropouts | 5 | 96 | threshold |
| *7* | 12 | 6 dropouts | 6 | 94.5 | threshold |
| *8* | 10 | 7 dropouts | 5 | 95.4 | threshold |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Network hyper-parameters*** | | | | | | | | | | | |
| *hyperparameters* | cov 1 | cov2 | cov 3 | cov 4 | cov 5 | cov 6 | cov 7 | cov 8 | cov 9 | cov 10 | fully connected |
| *number of filters* | 64 | 64 | 128 | 128 | 256 | 256 | 512 | 512 | 1024 | 1024 | n/a |
| *filter size* | 5 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | n/a |
| *stride* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | n/a |
| *zero padding* | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | n/a |
| *max pooling* | no | yes | no | yes | no | yes | no | yes | no | yes | n/a |
| *kernel size* | n/a | 2 | n/a | 2 | n/a | 2 | n/a | 2 | n/a | 2 | n/a |
| *stride* | n/a | 2 | n/a | 2 | n/a | 2 | n/a | 2 | n/a | 2 | n/a |
| *dropout* | no | no | 0.25 | no | 0.25 | no | 0.25 | no | 0.25 | no | 0.5 |
| *activation function* | ReLU | ReLU | ReLU | ReLU | ReLU | ReLU | ReLU | ReLU | ReLU | ReLU | ReLU |



Flow chart 1

|  |  |
| --- | --- |
| learning rate | adapting to the loss |
| **loss function** | Cross Entropy Loss |
| **Optimizer** | Adam |
| **batch size** | 300 |
| **Number of Cov layers** | 10 |

Reference

[1] A. Budhiraja, “Learning Less to Learn Better - Dropout in (Deep) Machine learning,” Medium, 15-Dec-2016. [Online]. Available: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5. [Accessed: 22-Mar-2018].

[2] 2018. [Online]. Available: https://www.researchgate.net/post/Could\_you\_give\_me\_some\_advices\_about\_how\_to\_improve\_the\_performance\_of\_Convolutional\_Neural\_Networks. [Accessed: 22- Mar- 2018]

[3] A. Dhingra, "Model Complexity-Accuracy Trade-off for a Convolutional Neural Network", Arxiv.org, 2018. [Online]. Available: https://arxiv.org/abs/1705.03338v1. [Accessed: 22- Mar- 2018].

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research, Jun. 2014.

[5] "Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f. [Accessed: 22- Mar- 2018].