ECE226 Project Report

Xinqiao Zhang

May 31 2020

1 Part1

Q1. Describe the classification task in a few sentences.

A classification task will classify things into different categories, for example, classifying cat picture and dog picture. A classification task has the goal to predict the class for a given unlabeled item. The class must be selected among a finite set of predefined classes.

Q2. Describe the data pre-processing in step 1. Explain what one-hot encoding is.

Three common data preprocessing steps are formatting, cleaning and sampling [1]. Step 1 use formatting and one-hot encoding is one of the method, a one-hot is a group of bits among which the legal combinations of values are only those with a single high '1' bit and all the others low '0' [2]. One-hot encoding is a process to convert integer variables to one-hot binary variables.

Q3. Write and explain the formula for the categorical cross entropy loss function.

$$C = -\frac{1}{n}\sum_{m}\ [y \ln a + (1-y)\ln(1-a))] \tag{1}$$

where n is the total number of items of training data, the sum is over all training inputs, x, and y is the corresponding desired output [3].This loss function can improve the learning slowdown issue un the beginning of iterations. The reason it can improve slowdown issue is that according to the partial derivative of the cross-entropy cost with respect to the weights (2)., The larger the error, the faster the neuron will learn.

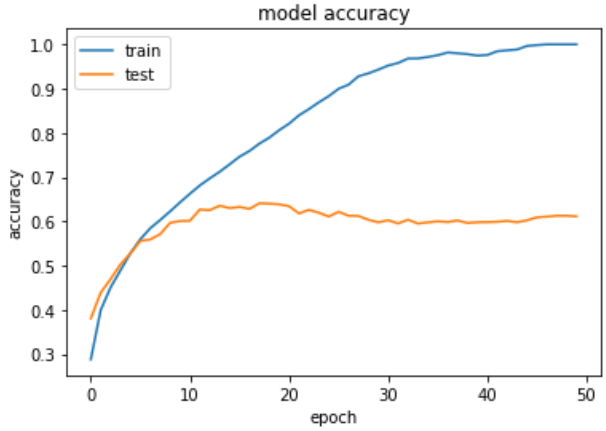
$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_{x} x_j(\sigma(z) - y) \tag{2}$$

Q4. Provide plots of training and test accuracies for steps 4-7. Each figure should contain training and test accuracy curves for one of the experiments. Comment on the effect of each step on the convergence of the model.

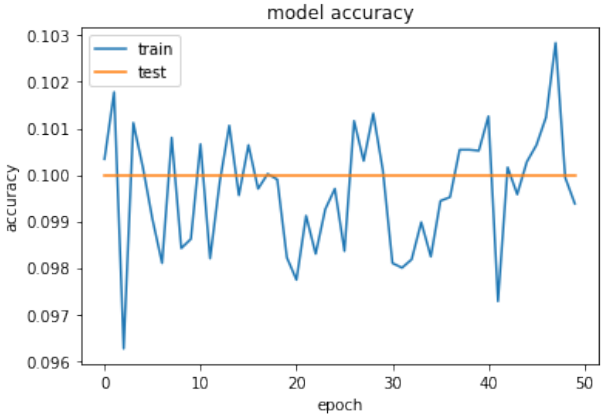
As we can see from Figure 1 below, the effect of each step on the convergence of the model is shown as follows: For step 4, $batchsize = 100$ makes the model converge. For step 5, $learningrate = 0.1$ can harm the convergence while $learningrate = 1$ can harm the convergence even worse. For step 6, 2x2 max-pooling layers helps with convergence. For step 7, use the "ImageDataGenerator" makes much better help on convergence. Therefore, apply ImageDataGenerator and set learning rate not big will make the model converge better and get a very good model.

Comment Step 7: on the effect of random cropping on the test accuracy. Which model is less overfitted?

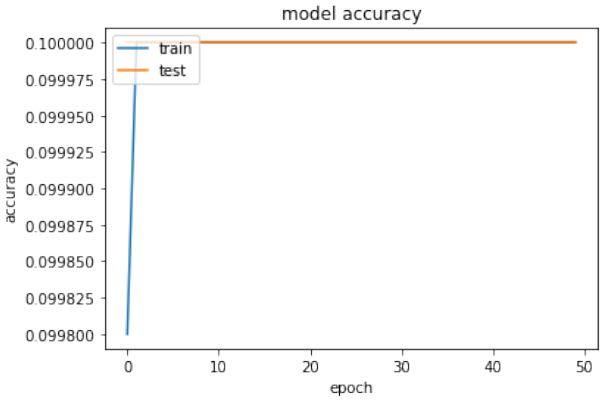
I think based on the plot below, the original model with $lr = 0.1$ is very random and the accuracy is pretty bad. But the new model with $Datagenerator$ and $lr = 0.1$ is less overfitted because the test accuracy is the highest among all the rest models.



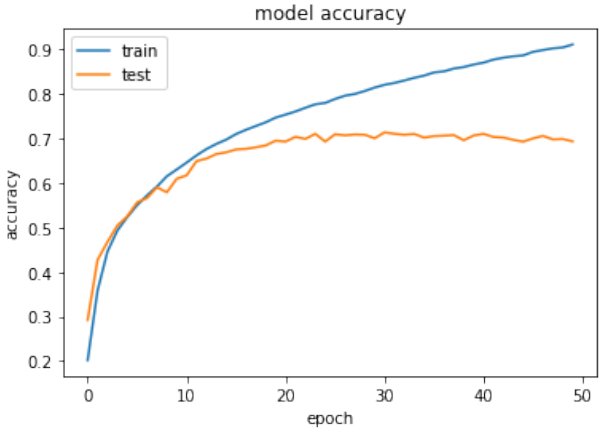
(a) Step4, batchsize =100, lr=0.001



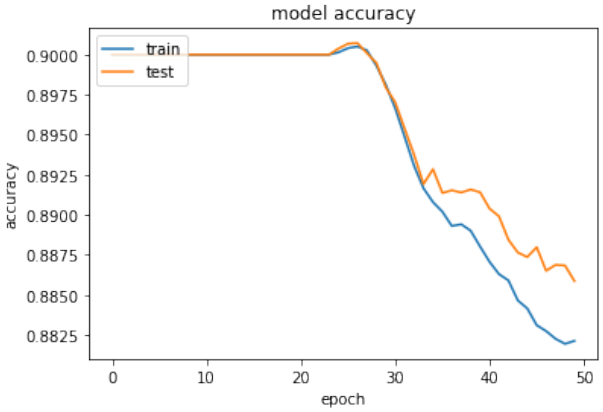
(b) Step5, lr=0.1



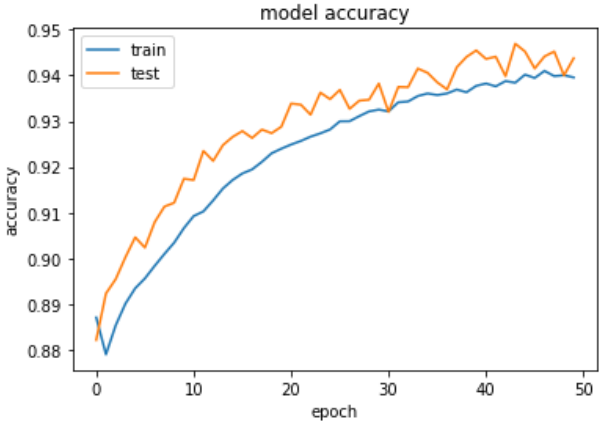
(c) Step5, lr=1



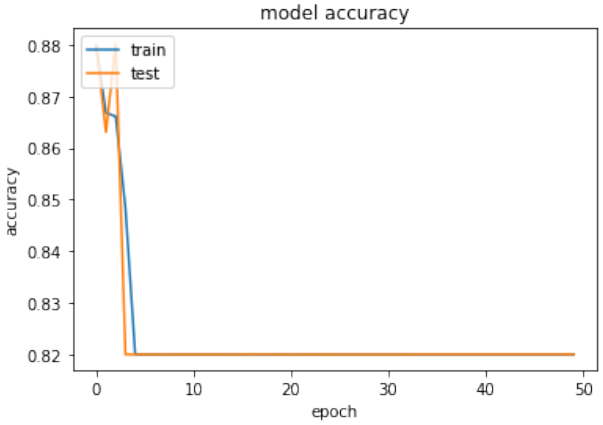
(d) Step6, add 2x2 max-pooling layers



(e) Step7,w/DataGenerator lr=0.001



(f) Step7,w/DataGenerator lr=0.1



(g) Step7,w/DataGenerator lr=1

2 Part2

Q1. Explain pruning and its benefits..

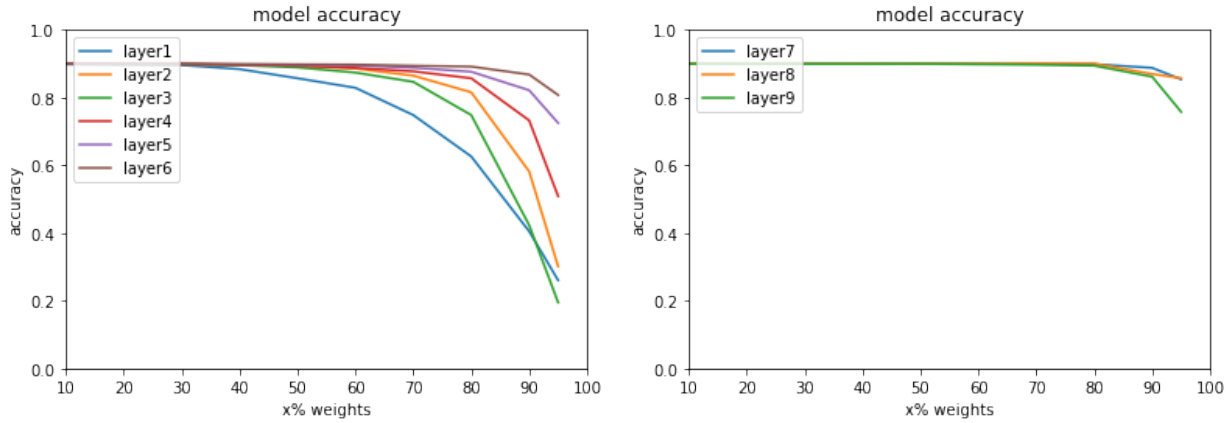
Pruning is a technique in machine learning and search algorithms that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting [4]. Especially for hardware implementation, pruning can take advantage of limited hardware IOT devices or other low-power devices. By applying pruning, it can both improve the speed and save power consumption, which is a really hot topic recently.

Q2. Explain the drawbacks of pruning. What are some challenges regarding performance acceleration of pruned models?

Drawbacks of pruning could be pruning may eliminate the interesting information which can lead to reducing the accuracy, which can cause model accuracy degradation.

Here are some challenges such as (1)There is no golden rule to measure which approach is the best. How to choose the proper method really depends on the applications and requirements . (2) Most of the current state-of-the-art approaches build on well-designed CNN models, which have limited freedom to change the configuration (e.g., network architectures, hyper-parameters) (3) Hardware constraints in various of small platforms (4) pruning channel is efficient but also challenging because removing channels might dramatically change the input of the following layer. (4) Methods of structural matrix and transferred convolutional filters impose prior human knowledge to the model, which could significantly affect the performance and stability. [5]. Those challenges are needed to improved or solved.

Q3.Comment on your results from step 4. This step does not retrain the model, so there is an accuracy drop. Do all the layers that you experimented with have the same effect on the overall accuracy?



(a) pruning layer 1-6 from 10% to 95%

(b) pruning layer 7-9 from 10% to 95%

Figure 2: model accuracy curves for part2.

As we can see in Figure 2, for conv layers, the first few layers drop the accuracy first as x increases and dense layers will not drop until reaches about 80% purning rate.

3 Part3

Q1. Describe the goal of the project.

As the topic shows, the main idea of this project is to learn some methods to optimize and accelerate the deep neural networks and make them compatible with the current various hardware platforms. I think the main idea of optimization and acceleration is with the development of all kinds of IOT devices, some reliable and low-power hardware platform is needed to do this stuff. And this is a very hot topic and it is very promising in the future.

Q2. Explain Tucker decomposition. Elaborate on how a convolution can be converted into subsequent convolutions using Tucker decomposition. Specifically, mention how the padding, stride, activation function, and bias term of each of the decomposed convolutions relate to the original non-decomposed convolution.

Tucker decomposition is a method that decomposes a tensor into a smaller core tensor and a set of matrices. Tucker decomposition is a higher order extension of the singular value decomposition (SVD) of matrix, in the perspective of computing the orthonormal spaces associated with the different modes of a tensor. It simultaneously analyzes mode-n matricizations of the original tensor, and merges them with the core tensor **??**.

As for Convolution layer, convolution layer weights are basically a 4-way tensor $W^{k*k*c*f}$ where k is the filter size, c is the number of input channels, and f is the number of output channels. By performing Tucker decomposition on W, there will subsequently be one core matrix and 4 matrices, effectively converting a convolution into four smaller convolutions.

$$W = core *_3 I *_4 O \tag{3}$$

where $core \in R^{k*k*R1*R2}$, $I \in R^{c*R1}$ and $O \in R^{R2*f}$. The ranks $R1$ and $R2$ can be modified depending on the desired compression of the weights. We need to note that to perform the three convolutions correctly, padding of the convolutional layers must be the same as the inputs. The first convolution layer will have a stride and kernel size of 1. In terms of weights, it will use I for the kernel and have no bias. The second convolution layer will use the *core* tensor as the kernel weights and zero bias. It will have the same stride, kernel size, and activation layer as the original convolution layer. As for the last layer, it will use the O tensor as the weights and the bias from the original convolution layer, the stride and kernel size should be set to 1.

Q3.Formulate the computational complexity of the decomposed layers based on the original filter W, the size of the input channels, and the decomposition ranks R1 and R2. Assume a stride of 1 and same padding for the original convolution layer. Comment how the decomposition can reduce the computational complexity.

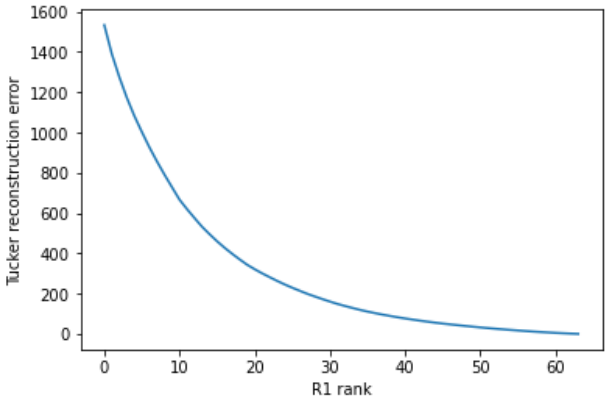
In general, the computational complexity of a convolutional layer is $k * k * c * f * m * n$ where m and n is the size of the input, assuming a stride of 1 and padding such that the output has the same size as the input. By performing Tucker decomposition on the weights, there will be more convolutional layers but a decrease in the number of computations. Specifically, it takes $m * n * c * R1$ for the first convolution layer, $m * n * k * k * R1 * R2$ for the second layer, and $*n * R2 * f$ for the last layer which comes out to be a total of $(m * n)(c * R1 + k * k * R1 * R2 + R2 * f)$.

$$\begin{aligned} ComplexityReduction &= \frac{k * k * c * f * m * n}{(m * n)(c * R1 + k * k * R1 * R2 + R2 * f)} \\ &= \frac{k * k * c * f}{c * R1 + k * k * R1 * R2 + R2 * f} \end{aligned} \tag{4}$$

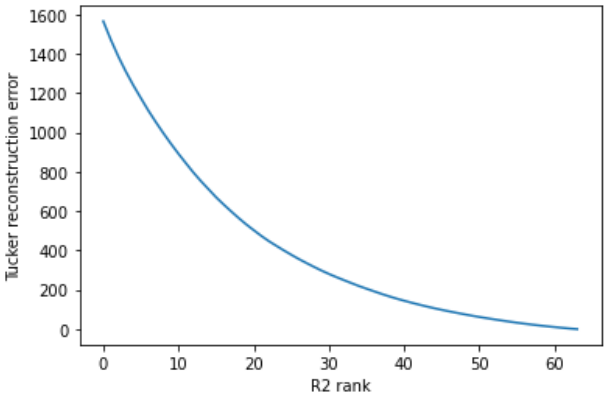
As Equation 4 shows, the complexity can be reduced based the value of R1 and R2, if we can find a proper value of R1 and R2 without much accuracy loss, it could be a very good method to reduce computational complexity.

Q4.Obtain the figures in Steps 1-3 of the project and explain the results.

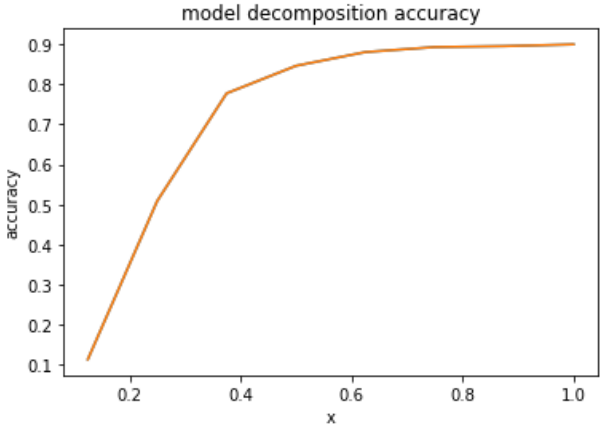
From the Figure 3, we can see that if we change either $R1$ while fix $R2$ or change $R2$ while fix $R1$, the tucker reconstruction error is decreasing as rank increase until the rank reaches 64. I think the higer $R1$ or $R2$, the less error it is and the less complexity reduction it will be. Also, as $R1$ and $R2$ increase, it will increase computational complexity.



(a) Change R1 while fix R2.



(b) Change R2 while fix R1.



(c) test accuracy of the network versus x.

Figure 3: model accuracy curves for part2.

Therefore, we need to find a good point to handle the trade-off between $R1/R2$ and accuracy. In order to do this, we have step3, which helps us find a good value point by tuning the value of $R1$ and $R2$, as the figure shows, we can see that if we set x to around 0.8, it is good enough to maintain the original accuracy while reducing some computational complexity.

References

[1] How to prepare data for machine learning. `https://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/m`. Accessed: 2020-05-31.

[2] David Money Harris, Sarah L Harris, Peter Prinz, and Tony Crawford. Digital design and computer architecture. 2019.

[3] Improving the way neural networks learn. `http://neuralnetworksanddeeplearning.com/chap3.html`. Accessed: 2020-05-31.

[4] Wikipedia contributors. Decision tree pruning — Wikipedia, the free encyclopedia, 2020. [Online; accessed 30-May-2020].

[5] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.