Convolutional Neural Network on CIRFAR-10

Yu-Chieh Chen
University of California, San Diego
La Jolla, California
yuc399@ucsd.edu

Ting-Yang Hung
University of California, San Diego
La Jolla, California
t6hung@ucsd.edu

ABSTRACT

Understanding big convolutional neural network(CNN) structures such as ResNet is an important topic in deep learning. By understanding how each layer from the most up to date CNN structure works, this knowledge would help people to develop an effective model to solve classification tasks. In our project, we utilized the structure of ResNet18 as a skeleton to fit our model with the CIFFAR-10 dataset. We want to find out whether utilizing the entire ResNet18 structure or to just utilize a portion of the Resnet18 structure gives the best performance on making predictions of unseen data. We hypothesize that utilizing the entire ResNet18 model will give the best performance overall. We fit the models with multiple different optimizers and evaluate each model based on the accuracy of validation set and test set. We find out that Adaptive Moment Estimation(Adam) is the best optimizer to fit the model. Then we proceed to dissect ResNet18 into 4 blocks, and fit a model on a different number of blocks with Adam optimizer. We discovered that the model trained with 2 blocks of ResNet18 gives the best testing accuracy: 80.25%. This result countered our first intuition. We suspect that the full structure of ResNet18 may be an overkill on a small dataset that leads to this result. Lastly, we develop our own model by first applying transfer learning to obtain two blocks of pre-trained parameters from ResNet18 with the latter portion being our own model architecture. The model has a test accuracy of 80.75%, which slightly out-performs the CNN architecture of 2 layers ResNet18 without transfer learning. This proves that transfer learning does increase the robustness of the model.

1 INTRODUCTION

In the past decades, Scientists had invested gigantic time to research on developing algorithms to allow machines to classify objects accurately just like humans. Although countless breakthroughs on classification tasks have been achieved by machine learning algorithms such as Decision Tree, Support Vector Machine, and Logistic Regression, these models are stagnant on accuracy when the dataset becomes large. In 1958, the first neural network was proposed by psychologist Frank Rosenblatt. The first neural network, called Perceptron, was inspired by the firing of neurons. The characteristics of neurons such as its threshold for firing a signal or receiving information through dendritic spines have been translated into mathematics to represent the learning process for the model. After years of research, the Convolutional Neural Network(CNN) has emerged that revolutionized the classification task by offering accurate prediction rate. Unlike the traditional Fully-Connected Neural Network, CNN aims to learn patterns from the data by performing convolution operation. CNN has performed exceptionally well on image classification. In our project, we utilize CNN to build our model for image classification tasks on CIFAR-10 Dataset, which includes 60000 training instances. Usually CNN performs well on large dataset(such as over one hundred thousand data) and does poorly on small dataset. We believe our dataset is of adequate size to train a CNN. We will be focusing on investigating the Residual Neural Net(ResNet) proposed by Kaiming He et al in 2015. We choose ResNet because it is the newest Neural Network Structure that best optimizes the loss on ImageNet Large Scale Visual Recognition Challenge(LCSVRC). (Figure 1)

After having examined the ResNet, we decided to investigate further on its first 18 layers, known as ResNet18. We noticed that inside each sequential block, there are two pairs of convolution layer and batch normalization layer followed by a ReLu activation function. We wonder how well this structure would achieve on the CIFAR-10 Dataset in terms of classification accuracy.

We experiment the training on different optimizers: SGD, Adam, Adamax, ASGD, Adadelta, AdaGrad, and RMSprop. In addition, we dissect ResNet18 into four blocks and train the model with different numbers of blocks. We experiment model training by reducing the number of blocks to combat the concern of overfitting . Finally, we experiment on transfer learning by extracting the pre-trained weights from ResNet18 and combine it with our own model architecture. We are interested in whether or not feature extractions will offer a better accuracy compared to our best model trained with ResNet18 structure without transfer learning.

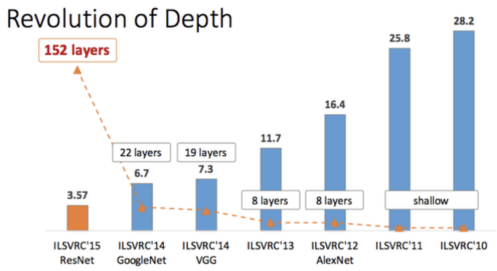

Figure 1: ImageNet Classification top-5 error (%)

2 DATASET OVERVIEW

We use the CIFAR-10 dataset[3], a subset of tiny images dataset, collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The dataset consists of 60000 training instances and 10000 testing instances. We obtain 10000 validation instances by randomly sampling from the training instances. Therefore, we have 50000 training instances to train our model. There are a total of 10 classes inside the dataset, and each image is of dimension 32 x 32. In order to apply ResNet18 to fit the model, we made a slight modification on the very last linear layer of ResNet18 to have 10 outputs to align with the number of classes on our dataset.

2.1 Training Set

We have 50000 training instances to fit the model, with 5000 images for each class. The class label is given as numerical representation(0 to 9) and can be loaded from pytorch's built-in package.

2.2 Validation Set

Since the dataset itself does not contain a validation set, we randomly sampled 10000 instances from the training set as our validation with each class containing 1000 images. The validation set allows us to make sanity checks on whether or not our model has performed well for each epoch training.

2.3 Testing Set

There are a total of 10000 testing instances with 1000 images per class. We based our final evaluation of the model on its performance on the test set.

2.4 Difficulties

Originally, we had considered using a Tiny image dataset to fit our model. However, we noticed that it includes an enormous amount of class instances(roughly around 200 or more) and that it does not include class labels for the test set. Therefore, we decided to proceed with model training with the CIFAR-10 dataset, which in our opinion is also a good quality dataset for deep learning.

3 DATA PREPROCESS

We downloaded the CIFAR-10 dataset from torchvision.datasets.CIFAR10 [3]. Then, we simply load the data with Dataloader and normalize the weights of RGB color and its standard deviation with the following values: (0.5, 0.5, 0.5), (0.5, 0.5, 0.5). In addition, we randomly rotate images by 20 degrees and introduce 50% of chance to flip the image horizontally. Normalization helps to reduce bias on large pixel values; The rotation and flip helps to combat rotation variant issues. This preprocessing aims to increase the robustness of the model on images that have unequal distribution of pixel values and images that are rotated and flipped.

4 METHOD DESCRIPTION

4.1 ResNet18 Overview

The Residual Neural Net(ResNet) is designed to speed up the training process and to achieve effective performance. People have discovered that stacking multiple convolution layers in general boosts the performance of the model. In addition, ResNet solves the vanishing gradient issue by utilizing identity map and ReLu activation function. According to Kaiming He, "Residual neural networks do this by utilizing skip connections,

or shortcuts to jump over some layers." This skipping over layers help to avoid vanishing gradients issue by reusing activations from a previous layer until the adjacent layer learns its weights [1]. In this way, the network contains less layers in the initial training stage and speeds up the training process. The below plots are illustration of ResNet structure and its skip connection features:

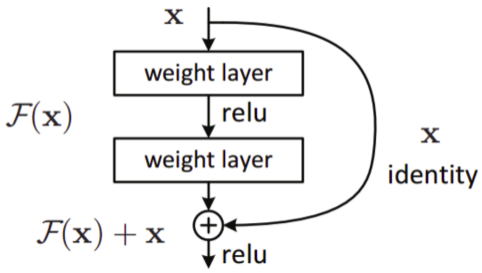

Figure 2: ResNet18 Skip Connection

4.2 Number of Blocks

The entire ResNet18 architecture contains around 11.2 million trainable parameters. The large number of parameters seem to accommodate large dataset. However, the size of our CIFAR-10 dataset is relatively small compared to the dataset that ResNet18 had trained on. We could see that the number of parameters decreases exponentially as the number of blocks decreases as illustrated in the plot below.

Blocks	Number of Parameters
4	11181642
3	2785354
2	**684362**
1	158154

4.3 Optimizers

4.3.1 RMSProp. RMSProp is one of the popular optimizers in deep learning. It takes into account the past gradients and the current gradient. Normally, P is set to 0.9. This means that it is weighting the present more than it weighs the past. The mechanism helps to control the step size of arriving at the minima. In simple manner, RMSProp constantly scales the learning rate to avoid exponentially decaying gradients. [4]

$$For\ each\ Parameter\ w^j$$
$$(j\ subscript\ dropped\ for\ clarity)$$
$$\nu_t = \rho\nu_{t-1} + (1-\rho) * g_t^2$$
$$\Delta\omega_t = -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

$\eta : Initial\ Learning\ rate$
$\nu_t : Exponential\ Average\ of\ squares\ of\ gradients$
$g_t : Gradient\ at\ time\ t\ along\ \omega^j$

Figure 3: RMSProp Algorithm

4.3.2 Adaptive Moment Estimation (Adam). Adam is by far one of the most effective optimizers in deep learning. However, some researchers have discovered that Adam may perform worse than Stochastic Gradient Descent(SGD) in non-convex optimization of neural networks. Adam can be viewed as a combination of RMSProp and SGD with momentum. Therefore, Adam also helps to combat the exponentially decaying gradient issue while maintaining an appropriate step size to approach the optimal parameters.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figure 4: Adam Algorithm

4.4 Transfer Learning

Since we are utilizing ResNet18, we would like to observe the effect of using the pre-trained weights to the performance of the model. We select the first two blocks of ResNet18 as feature extractions. Then, we attach our model architecture right after the two blocks. We disable the parameter updates for the ResNet18 layers so that the training will only update the parameters defined in our model. One reason that we only do feature extraction on the first two layers is that we know at the

very initial training state, only some coarse patterns will be learned such as horizontal and vertical stripes. Therefore, we hope to obtain coarse patterns before training our model to give our model a slight jump at the starting line.

5 EXPERIMENT/IMPLEMENTATION

5.1 Training Design

In all of the CNN models training, we set the batch size to be 32 and the number of epochs to be 12. That is, the model is trained with 32 instances at a time and the training process goes over the dataset for 12 rounds. After the training of each epoch, the evaluation mode on the validation set is performed to better observe the model's robustness to perform classification tasks. The training accuracy, training loss, validation accuracy, and validation loss are recorded for every epoch. Finally, we evaluate the model using the test set and obtain a test accuracy.

5.2 Comparison of Different Optimizer

As we see in table 1, figure 5, and figure 6, Adadelta performs the worst in training, validation, and testing dataset. In all the optimizers we had tested, Adam performs the best in training, validation, and testing dataset. We will use Adam as our optimizer to do further research.

Optimizer	Training	Validation	Testing
SGD_lr_0. 001 m_0.9	74.9	68.06	70.48
SGD_lr_0. 001 m_0.0	57.66	55.01	56.6
Adam	**83.04**	**77.55**	**79.32**
Adamax	81.92	76.01	77.74
ASGD	57.16	53.71	55.41
Adadelta	41.58	39.90	41.68
Adagrad	62.44	58.04	59.91
RMSprop lr_0.001_m_0.0	82.16	77.08	78.18
RMSprop lr_0.001_m_0.9	77.50	72.82	75.01

Table 1: comparison of different optimizer

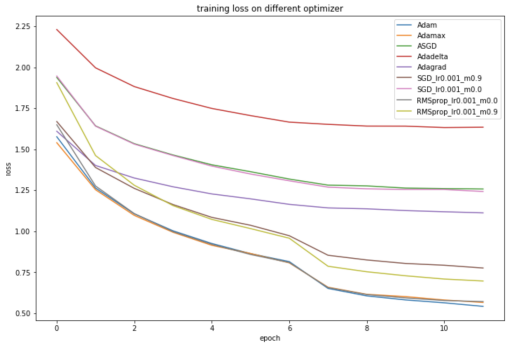

Figure 5: Training Loss

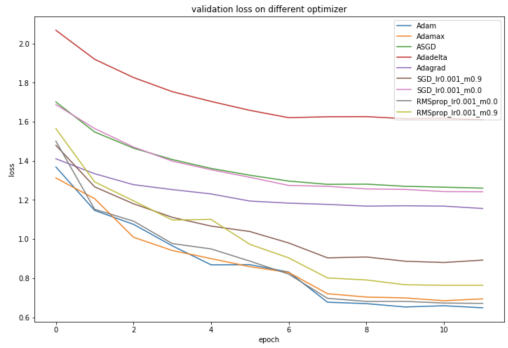

Figure 6: Validation Loss

5.3 Comparison of Different Blocks

In this section, we modify the original Resnet18 model [2]. The original ResNet18 model has 4 blocks with training accuracy of 83.04% and validation accuracy of 77.55%. Model trained with 2 blocks of ResNet18 has the highest testing accuracy. However, models trained with 2 blocks and 3 blocks do not differ too much in the validation and testing accuracy.

Blocks	Training	Validation	Testing
4	83.04	77.55	79.32
3	**84.40**	**78.04**	79.59
2	82.97	77.86	**80.25**
1	77.02	74.45	76.91

5.4 Transfer Learning on 2-Layer ResNet18

Based on the result in 5.3, we decide to perform transfer learning by utilizing the first 2 blocks from ResNet18 because its testing accuracy is the highest. Our model has

a total of 9 layers: one block of two convolution operations followed by a Relu activation function and a batch normalization. Finally, an average pooling followed by 2 linear functions. The accuracy on the training set, validation set, and testing set is 81.15%, 78.03%, and 80.75% respectively. The testing accuracy is slightly better than the best model that we obtained from 5.3.

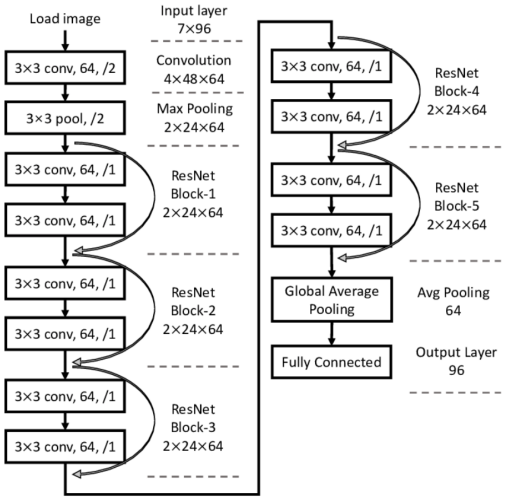

Figure 7: Original ResNet18 Structure

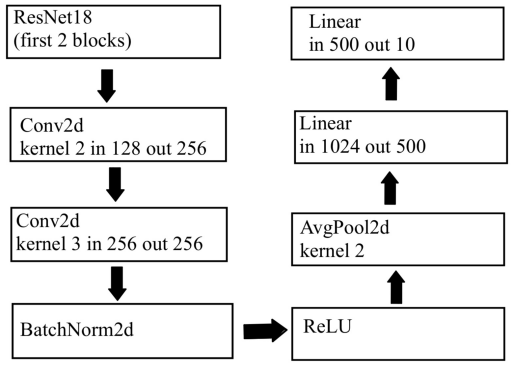

Figure 8: Transfer Learning Model Structure

6 DISCUSSION

In 5.2, we had compared models trained with different optimizers using the full ResNet18 as the model architecture. From our finding, Adam is the best optimizer. This aligns with our original conjecture because Adam can be viewed as a hybrid of momentum-based descent and RMSProp that helps to combat exponential gradient decay issues. In 5.2, we obtained that the model trained with 2 blocks of ResNet18 with Adam optimizer performs the best on testing accuracy: 80.25%. This result confirms our intuition because we don't think that the CIFAR-10 dataset requires a large number of parameters to be learned. The models trained with 4 blocks and 3 blocks seem to experience overfitting because the gap between the training accuracy and testing accuracy is apparent. Lastly in 5.3, the model utilizing transfer learning of 2 blocks ResNet18 and our own model architecture with Adam optimizer out-performs all models that we trained before. We think using the pre-trained weights puts the model further ahead of the starting line by offering coarse simple pattern identification such as detecting horizontal and vertical stripes. In addition, the model with transfer learning has an even smaller gap between training accuracy and testing accuracy. This means its training process captures the unseen data well.

7 LIMITATION

The data preprocessing could have been better. Originally, we were thinking of removing the background of the images to minimize the noise using Discrete Fourier Transform(DFT) before training the model. However, we noticed that the images in the CIFAR-10 dataset are too small(only 32 x 32) and have low resolution. Therefore, we were not able to really distinguish noise from the real object using DFT. We also tried the Haar Transformation but it did not work well as expected. Moreover, we feel we should have trained the model with more epochs. However, with a limited number of GPUs, increasing the epochs means increasing the training time. Due to time limitations, we were not really able to do it so we limited the training to only 12 epochs. Otherwise, we think everything else went smooth.

8 BONUS POINTS

8.1 Transfer Learning

We did some research on trying to understand how transfer learning works. We also tried to understand why Resnet is by far the most effective model in the LCSVRC.

8.2 Data Preprocessing

We tried multiple approaches to reduce the noise of the images before training. We introduced flips and rotations to combat rotation variant issues. We also tried to apply DFT and Haar Transform with filtering to reduce the noise of the images. Unfortunately, those pre-processing didn't work as expected. However, we spent a huge amount of time figuring out this. We feel we deserve some rewards for our efforts.

REFERENCES

[1] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385* (2015).

[2] Nathan Inkawhich. *Finetuning TorchVision Models*. URL: https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html. (accessed: 06.59.2020).

[3] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[4] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).