Mini Project - ResNet under 5M

Sai Krishna Prathapaneni,[1] **Shvejan Shashank** [2]

[1,2] Department of ECE, New York University 1
sp7238@nyu.edu, ssm10076@nyu.edu, https://github.com/saikrishna-prathapaneni/resnet5M

Abstract

The project aims to design a neural network architecture for image classification on the CIFAR-10 dataset that contains less than 5 million parameters. The goal is to create a model that can achieve high accuracy while being computationally efficient in this case ResNet-based model under 5M parameters. The project will involve researching and experimenting with different architectural design choices, such as the number of layers, filters, and kernel sizes. The model's performance will be evaluated based on its accuracy, training and inference times, and parameter count. The final outcome will be a compact and efficient neural network architecture for image classification on the CIFAR-10 dataset, which reaches an accuracy of 86.6%. It is also shown that at certain parameters widening of the model in addition to the depth of a model can also perform better at approximately the same parameter count. Codebase is implemented with Pytorch (Paszke et al. 2017)

Introduction

Deep Convolutions tend to perform better when considered with an increase in depth and also an increase in width. But as the depth is increased we observe a gradient vanishing problem arising and to resolve that residual blocks as shown in 1 are introduced by which backpropagation reaches the lower layers thereby updating the values significantly. Below are some of the hyperparameters considered for the model tuning in addition to the following parameters, activations and optimizers are also changed accordingly.

- N : Number of layers in the model.
- B_i : Number of residual blocks in ith layer.
- C_i : the number of channels in the ith layer.
- F_i : the filter size in the ith layer
- K_i : the kernel size in the ith skip connection
- P_i : the pool size in the average pool layer,

A series of experiments are conducted using ResNet models trained on the CIFAR10 dataset. In each model iteration, we identified and addressed architecture shortcomings in order to improve performance. After multiple rounds of experimentation with different architectures, our res_512_d model stood out as the top performer, with Leaky ReLU activation function, SGD optimizer, and Cosine Annealing LR scheduler with dropout. Over a span of 200 epochs, this model achieved an accuracy rate of 86.66%.

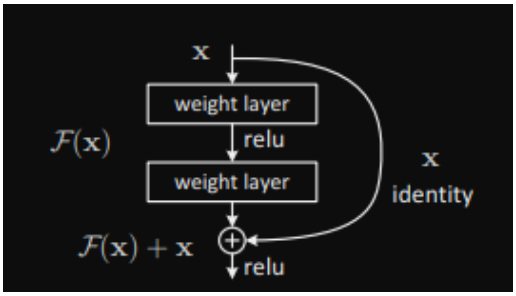

Figure 1: Basic Residual block in ResNet architecture (He et al. 2015)

Architectural changes

On a broader level, three modifications to the model architecture are explored. In one iteration, the width of the architecture was considered, while in the following iteration, the depth of the model is concentrated over the other three. In the last iteration, Residual block changes have been introduced to check the effects as seen in Figure 3. All of the models considered are having less than 5M parameters in total, Table 2 shows the different parameter considerations.

Initially, we started with the SGD optimizer to test our models with a learning rate of 0.1 and cosine annealing(Loshchilov and Hutter 2017) as the learning rate scheduler. We applied simple data argumentations like random crop, horizontal flip, rotation, and normalization. We trained all of our models for 200 epochs with a batch size of 128. The table 2 below summarizes the parameters and configurations of our models.

Note:

1. In the ResBlock model, we experimented by making modifications to the original residual block, in this architecture, we added an extra $Relu \rightarrow conv \rightarrow BN$ as shown in Fig 7 but made sure that the parameters are under 5M as shown in Table 1.

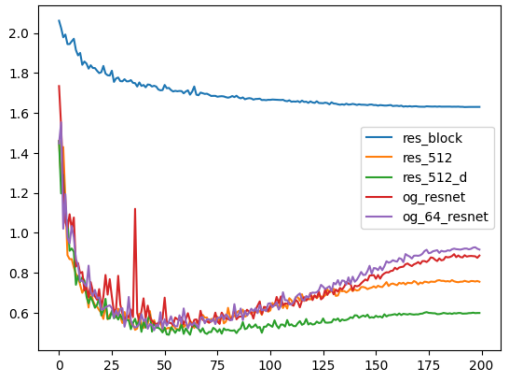

Figure 2: loss vs epochs

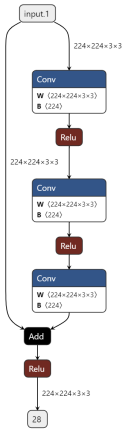

Figure 3: Modified Resblock

2. res_512 model achieves 86% accuracy on the test but over-fits on train accuracy to over 98%.
3. res_512_d model achieves $\approx$ 86% but now the model train accuracy is confined to 86% with added Dropout layer in the Residual block.

Other architectural considerations:

we experimented with other architectures which are not conventional to residual networks. following are the results obtained with the changes.

- Added extra layer of Linear with 1024 units each with avg pool output size as with final output residual conv having 256 filters and avgpool output as (2,2) i.e P=2 which produced an accuracy of 75% and parameters $\approx 3.7M$.
- Added extra layer of the Linear layer with 2048 hidden units and avg pool output as (2,2) and residual layers final output channels as 512 that achieved an accuracy of 76% with train accuracy as 81%.
- added an extra layer block at the end of ResNet with $conv \rightarrow BN \rightarrow ReLU \rightarrow conv \rightarrow BN$ before avg pool without residual connections which produced an accuracy of 78%. after 100 epochs of training.

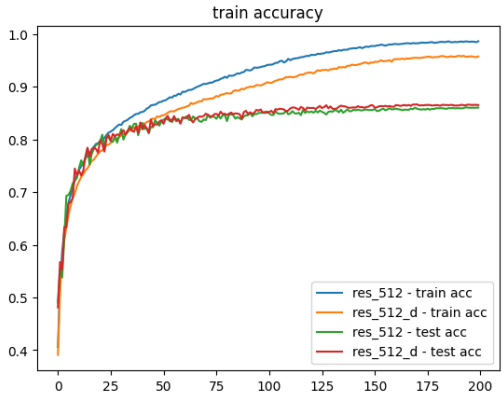

Figure 4: validation and train accuracy

The model with a concentration of Width res_512_d as our final model which has the 4.9M parameters approximately which performed the best among the architectural changes considered, Image augmentations, and learning rates, upon which we started training the model by changing the optimizer and Activations.

Optmizer

A series of experimentations were conducted by running our best performing model Res_512_d on 4 different optimizes, Adadelta (Zeiler 2012), Adam(Kingma and Ba 2017), RMSprop, and SGD for 50 epochs using ReLU activation and Cosine Annealing learning rate scheduler. The results are shown in the fig below. from the experimentations, It can be observed that SGD gave us the best results with 84% accuracy on 50 epochs and Adadelta came second with 82% accuracy, having the rest of the Augmentations, Activation as ReLU, learning rate as 0.1 and batch size as 128.

Activations

Next, the evaluation of the Res_512_d model is done by experimenting with four different activation functions: ReLU, GeLU (Hendrycks and Gimpel 2020), LeakyReLU, and Tanh. The model was trained for 50 epochs using a Cosine Annealing learning rate scheduler as shown in Table 3, having rest (Xu et al. 2015) of the parameters constant which is optimizer to be ReLU, batchsize 128, same Augmentations.

The results of our experiments are presented in the fig below. Our analysis revealed that the LeakyReLU activation function outperformed the other three functions, achieving an accuracy of 85.1%. The GeLU activation function came in second with an accuracy of 84.7% after 50 epochs.

Pros, Cons and Lessons Learned from Design Choices

- Increasing the nodes in a fully connected layer didn't add up to accuracy.
- Consequently nodes increment shall be directly affected by average pool size increment, which in this case is P=2.

Model	Params	N	B	C	F	P	Acc
og_64_resnet	4,997,802	3	4,5,3	64,128,256	3,3,3	1	85.87%
og_resnet	4,997,802	4	1,4,5,3	32,64,128,256	3,3,3,3	1	85.1%
res_512	4,961,610	3	3,6,1	64,128,512	3,3,3	1	86%
res_512_d	4,961,610	3	3,6,1	64,128,512	3,3,3	1	86.66%
ResBlock	4,979,498	4	1,4,3,1	32,64,128,256	3,3,3,3	1	83%

Table 1: Results of different architectures

Activation	Accuracy
ReLU	84.62%
LeakyReLU	85.1%
GeLU	84.73%
Tanh	78.74%

Table 2: Activation vs Accuracy

Optimizers	Accuracy
AdaDelta	82.85%
Adam	80.84%
RMSProp	78%
SGD	84.62%

Table 3: Optimizer vs Accuracy

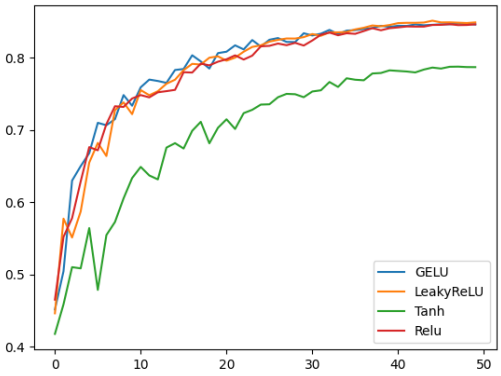

Figure 5: Activations vs Accuracy

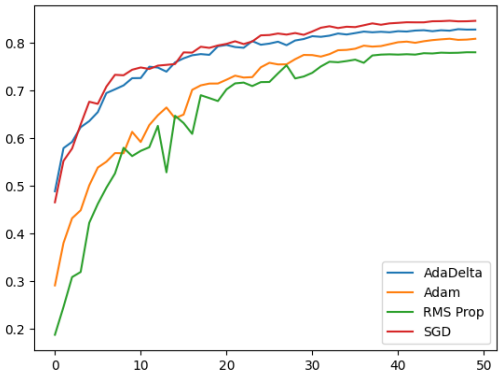

Figure 6: Optimizers vs accuracy on res_512_d

- Though the avg pool of 2 would give more generalization of filters and corresponding flattening, it increases the complexity of the fully connected layer(s).
- As shown in Fig7 addition of a $conv \rightarrow bn$ into residual block didn't add up to accuracy rather degraded the performance of the model can be inferred from 6.
- Tanh activation is wrost performing nonlinearity for the final model with 78%
- Both res_512 and res_512_d performed very similarly on the test data but as shown in Figure 4, adding dropouts helped the res_512_d model to generalize well over the training data and prevented it from overfitting.
- While cropping Augmentation improves the model generalization, center cropping makes the model worse, since the some of the distinctive features that a model should understand might cutoff from the image.
- While having $\approx constantparameter$ count for the models of res_512_d which has 512 layers in the last layer and og_64_resnet which has 256 features on the last layer and comparitively more depth, res_512_d model outperformed the depth model, though not by significant margins, gives us a thought to explore more in to considerations of wider models in parameter constrained systems.
- In longer training time or epochs depth significant model overfit more than the wider significant models.

Results

Table 2 displays the test accuracies of the res_512_d model with different activations. From the results, we observe that Leaky ReLU has the highest test accuracy of 85.1%

Table 3 displays the test accuracies of the res_512_d model with different optimizers. From these results, we observe that SGD optimizer consistently gives us the best results with an accuracy of 84.62%

Table 1 displays the model architectures' parameter sizes, configurations, and accuracies after 200 epochs of training. from these results, we observed that the res_512_d model has the highest test accuracy with 86%

Figure 1 shows the structure of a simple residual block

Figure 2 shows loss vs epochs for all the models which we considered in this project. From the graph, we can observe that res_512_d gives the best result.

Figure 3 shows the modified residual block we implemented in our ResBlock model but this architectural change did not achieve the best results.

Figure 4 graph shows the comparison of test and train ac-

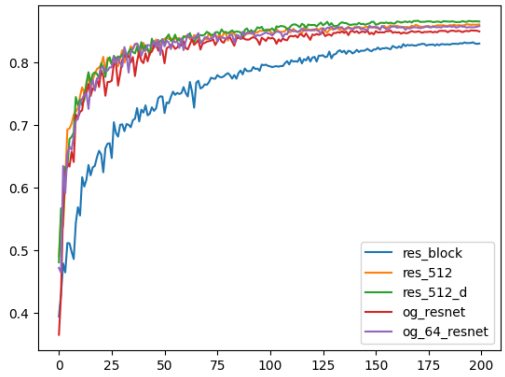

Figure 7: models vs accuracy

curacies of the two of our best performing models res_512 and res_512_d. from the graph, though both models achieved similar test accuracies, the res_512_d model with the dropout layer better generalizes the data with slightly lower train accuracy.

Figure 5 graph shows the accuracies of different activation functions from which we can observe that Leaky Relu gives us the best results

Figure 6 graph shows the comparison of different optimisers implemented on the res_512_d model. From the results, we observe that SGD optimizer gave us the best results.

Figure 7 graph shows the comparison of accuracies of different models which we considered in this project and the res_512_d model gave us the best results

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Hendrycks, D.; and Gimpel, K. 2020. Gaussian Error Linear Units (GELUs). arXiv:1606.08415.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. arXiv:1608.03983.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch.

Xu, B.; Wang, N.; Chen, T.; and Li, M. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv:1505.00853.

Zeiler, M. D. 2012. ADADELTA: An Adaptive Learning Rate Method. arXiv:1212.5701.