ResNet on Miniplaces Dataset

Jeffrey Hu
MIT
77 Massachusetts Ave, Cambridge, MA
hujh@mit.edu
Team Name: Procrastination

Abstract

Deep neural networks have achieved impressive results at image classification tasks. These techniques however require immense amount of training data. In this project, we attempt to achieve good levels of performance with a fraction of the data. The MiniPlaces dataset we use contains only 100,000 images. We implement ResNets of varying depth to see what architecture works best and combat overfitting with heavy data augmentation. We then explore ways to leverage pixel-level object annotations to further improve classification accuracy.

Figure 1. Sample images from the MiniPlaces dataset.

1. Introduction

Image classification is the classic computer vision task of classifying images into categories. Since 2010, the ImageNet project has hosted the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) to tackle this problem. Back then, a good classification error was around 25%. In 2012, the breakthrough paper Alexnet[1] brought this down to 16%. It was the first model to use a convolutional neural network (CNN) and one of the first demonstrations of the power of neural networks.

Since then, networks have gotten better but more complex. AlexNet contained 8 layers. VGG19[3] and GoogleNet[4] further improved performance and contained 19 and 22 layers respectively. In 2015, ResNet[2] became the state-of-the-art by achieving a classification error of 3%. This is par with human judges. It however used a whopping 152 layers and trained with 10 million images. In this project, we will primarily focus on tuning a smaller version of ResNet to a smaller dataset. We will see what architectures work best and discuss the challenges associated with training on a small dataset.

1.1. MiniPlaces

The data for this project comes from the Places2[5] dataset which contains over 10 million images belonging to 400+ categories. Here, we use a subset of this data. The MiniPlaces dataset contains 100,000 images for training, 10,000 images for validation, and 10,000 images for testing, coming from 100 categories. The images have been resized to 128x128 (from 512x512) for better manageability and faster training.

Also, some but not all of the images also have object annotations provided. These come in a separate file as a list of polygons. In Section 5, we discuss strategies for incorporating this data into our training process.

2. Related Work

The current state-of-the-art for image classification are deep residual networks called ResNets. These networks are effective largely because to their immense depth. Researchers have known for years that greater depth gives networks greater representational power. This is because in CNNs, lower layers tend to learn local features like edges and corners while higher layers learn larger concepts like faces, cars, and trees. More layers allow the higher layers build off of the activations of the lower layers and learn more complex concepts.

Previously, extremely deep networks were impossible to

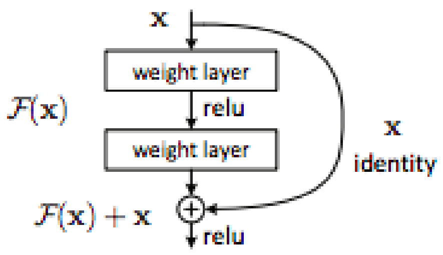
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block

train due to the vanishing gradient problem. Backpropagation over large numbers of layers chained multiplication operations together in such a way that gradients tended to zero. This resulted in inefficient learning. ResNet solved this problem by implementing skip connections. Skip connections allow gradients to propogate both through and around convolutional layers. This allows information from the gradients to travel more easily through the entirety of the network and results in more effective learning. Then, ResNets achieve their results from their sheer depth and representational power.

In this project, we choose to focus on ResNets because they are the state of the art and contain few parameters to optimize. The principles behind skip connections and residual learning is very general so we expect them to work just as well on small datasets.

3. Technical Approach

Since the goal of this work is to see how effectively we can learn from a small dataset, we train all networks from scratch, exclusively on MiniPlaces data. Our framework of choice is Keras backed in Tensorflow. From the original paper, there are 5 ResNet architectures to choose from: ResNet18, ResNet34, ResNet50, ResNet101, and Resnet152. Since we have so little data, we expect to need only a small network. Thus, we implemented ResNet18, ResNet34, and ResNet50.

During training, we prepared all data on the fly. Preprocessing all the images and their augmentations was not practical due to the amount of data augmentation we had prepared. To reduce the amount of time the program spends at CPU and better use GPU resources, we multithreaded our program and get multiple CPUs to support a single run. As a result, less than 5% of the training time is spent preparing the data. Training times with and without data augmentations are similar.

For ResNet18 and ResNet34, we used a batch size of 64. For ResNet50, we used a batch size of 32 since 64 did not fit on our GPU. Every network was trained until convergence, which was at least 100 epochs. Each epoch consists of 2000 iterations. Our final network was a ResNet50 trained for 120 epochs or 33 hours.

ResNet18	ResNet34	ResNet50
300s	500s	1000s

Table 1. Time required to train each epoch.

4. Experiments

4.1. Baseline

Keras comes with an implemented version of ResNet50. We choose to use this as our baseline. With no data augmentation, the training and validation errors quickly diverge. The training error heads to zero while the validation error remains at around 75%.

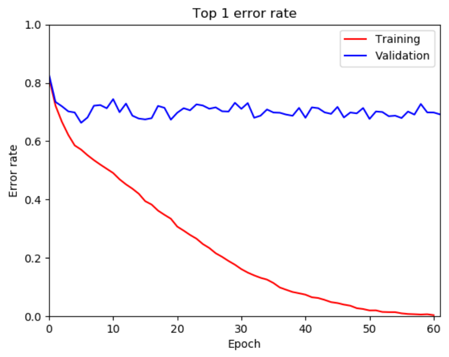

Figure 3. Training and validation error of baseline model.

4.2. Optimal

Before we improve on the baseline with data augmentation, we would like to get an idea of how well we can possible do. Since we are working with a small dataset, we do not expect to outperform networks trained on larger datasets. The features we extract will definitely be of lower quality those extracted by a network trained on a larger dataset. Thus, finetuning a network pretrained on a larger dataset gives us an upper bound on our performance.

With the same ResNet50 in Keras, we import weights from ImageNet with a single line of code. Finetuning this model with no data augmentation, we achieve a validation Top1 error rate of 43% within one epoch. This is significantly better than our baseline. The validation error gradually increases as our network overfits to the training set.

4.3. Data Augmentations

With the training error approaching 0%, it is clear the key to improving performance is utilizing a large amount of data augmentation. In total, we implemented 8 types of data augmentation. They are as follows:

Flip: Randomly flip image with probability 0.5.
Crop: Randomly crop the image to 80% its size.
Shift: Shift image in any direction by 20 pixels.
Rotate: Rotate image between +90 and -90 degrees.
Shear: Slant image by between +30 and -30 degrees.
Brightness: Randomly change the brightness.
Contrast: Randomly change the contrast.
Drop pixels: Randomly set 10% of the pixels to a random color.

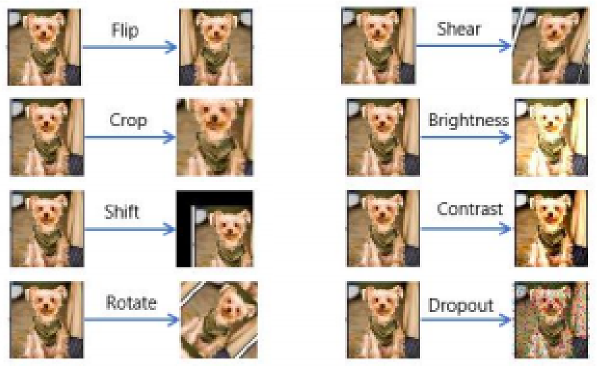

Figure 4. Data augmentations visualized.

For all networks, the error rate decreased significantly. Looking at ResNet50, the top1 error rate went from 75% to 63%, a 12% improvement over the baseline. From here on out, all networks are trained with the full suite of data augmentations.

4.4. Varying Depth

Smaller ResNets have less representational capacity. This makes them less powerful but also less prone to overfitting. Since we do not have much training data, a smaller network may outperform a larger overfitted network. Here, we find the best performing ResNet is ResNet34.

	Top1 error	Top5 error
ResNet18	0.6368	0.331
ResNet34	**0.6342**	**0.3234**
ResNet50	0.6542	0.3354

Table 2. Validation error rates of ResNets trained with data augmentation.

4.5. Final Network

The best performing network we found is actually slightly different from the ResNets described above. It is a ResNet50 with a larger starting image, 224x224 instead of 128x128. We tested this size because it is the smallest starting shape accepted by the default implementation of ResNet in Keras. For training, we simply resized the image to be larger before doing all the data augmentations. Inexplicably, this network performs better than all the other ResNets. It achieves top1 validation error of 0.5504 and a top5 error of 0.2516.

This result may be because of the larger layers it has. At every layer, this ResNet50 is four times as larger as the other ResNet50. Pooling halves the feature map dimensions of the later layers. It is possible that the pooling in the smaller ResNet50 may collapse the information too far, thus resulting in inferior performance. Unfortunately, due to the time constraints on this project, we were unable to investigate this further.

	Input size	Top1 error	Top5 error
ResNet50	128x128	0.6368	0.331
ResNet50	224x224	**0.5504**	**0.2516**
Optimal*		0.4323	0.1553

Table 3. Validation error rates of ResNets with differing input sizes. *Pretrained weights from ImageNet

5. Incorporating Object Annotations

As part to the Miniplaces challenge, object annotations are provided for a subset of the images. This data is highly informative because the objects within a scene are highly correlated with the category of the scene. A network that successful utilizes this information could make predictions based on crisp segmentations instead of blurry 128x128 images.

5.1. Model

We incorporate the object annotations by first converting them into binary masks. For K object categories, we generate Kbinary masks and stack them in the channel dimension to get an input with shape HxWxK. If no object annotation is present for some category, that slice contains only zeros.

We then use a two-head ResNet architecture for classification. The idea is to project the image and the binary masks into the same feature space. Working with ResNet50, we take two copies of the first 10 layers of ResNet to form two heads. One converts the image into a feature map of size 32x32x128 while the other converts the binary mask input. Then, we concatenate the two maps and continue through the rest of the ResNet with twice as many channels. Note the first layer of the binary mask head maps K channels to 50 channels while the image head maps 3 channels to 50 channels.

5.2. Results

Unfortunately, the effect of this new binary mask module on performance is negligible. As only 3% of the training and validation images have annotations, there may not have been enough data to learn something useful in the second head. There may have been too many parameters for the amount of segmentation data.

Given more time, we might try something simpler like a single vector saying whether an object annotation category is present. This would be incorporated with a fully connected layer towards the end of the network. Another idea is compress the K channels of the binary masks into smaller N channels where the values in the N channels represent categories are equidistant from each other. These N channels could then be concatenate with the original 3 RGB channels and run through the original ResNet. All in all though, the limited amount of segmentation data makes any approach difficult.

6. Conclusion

In this project, we attempt to get good performance with ResNets on a small dataset. With our baseline, we got 75% error rate. At the same time, our training error rate went to zero. To combat this immense overfitting to the data, we implemented several functions for data augmentation. After this, all networks saw a large boost in accuracy. We then tested ResNets of various depths but found they all had similar performance. Lastly, we tried to incorporate object annotations into the classification task but were largely unsuccessful.

References

[1] I. S. A. Krizhevsky and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[5] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.