The effect of different augmentation techniques on the cifar10 dataset in a deep convolutional neural network

Viktor Gummesson
vgum@kth.se

KTH Royal Institute of Technology

Abstract. This project investigate the effect that data augmentation have in a convolution network trained on the Cifar10 dataset. Five different augmentation techniques were experimented with and it was shown that augmentation and the number of augmentations used plays a big role to reduce overfitting. The best scoring network used two augmentation techniques, random flip left to right and random rotation, which scored an accuracy on the validation set of $\approx 82\%$. An observation that augmentation could possibly have a negative effect on how fast the network learns were also made, further research would be needed.

Keywords: CNN, Data augmentation, Deep neural networks, Image classification, Tensorflow, Tflearn

1 Introduction

In any type of AI, machine learning one of the most important aspect is how good is how good a model generalizes. With generalization we refer to how well the concepts learned by the model during training apply to specific examples not seen by the model when it was learning. I.e. how the model performs in practice. The goal of a good model is to generalize well from the training data to any data from its problem domain.

When evaluating how well a model learns and generalizes to new data we often look at overfitting and underfitting. Overfitting is when a model models the training data to well. This means that noise in the training data have been picked up by the model during training and learned by the model as concepts. This becomes problematic due to the fact that these concepts do not apply to new data and worsen the models ability to generalize.

One way to mitigate the effect of overfitting is called data augmentation. This is the technique we focus on in this report and take a look on how it affect a concrete deep convolutional neural network.

1.1 Problem formulation

Given a set of augmentation techniques, how does different subsets of those effect a deep convolutional network on the Cifar10[1] data set. Do more augmentation always mean less overfitting? Are some combination of augmentation techniques better suited then other?

2 Background

2.1 Convolutional neural networks

A convolutional network allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network[2].

The architecture in convolutionals is a combination of three ideas that ensure some degree of shift and distortion invariance. These are local receptive fields, shared weights, or weight replication, and sometimes spatial or temporal subsampling. A typical convolutional network, LeNet-5 / LeCun 1990, is shown in figure 1, it was used to recognizing handwritten characters. The input to the network consist of recived images of characters that are approximately size-normalized and centered[2].

Each unit in a layer recives its input from the previous layer which is a set of units located in a small neighborhood. This idea of connecting units to local receptive fields on the input goes back to the Perceptron algorithm and was almost simultaneous with Hubel ans Wiesel's discovery of locally-sensitive, where they found orientation-selective neurons in the cat's visual system[3].

With these local receptive fields the neurons can learn to extract basic visual features such as edges, end-points corners, etc. Then these features are combined by the higher layers to detect higher-order features. It have been showed that these basic features detectors that are helpful on one part of an image are likely to be useful across the entire image[3]. This fact can be used by setting a set of units, whose receptive fields are located at different places on the image, to have identical weights vectors. Units in a layer are organized in planes within which all the units share the same set of weights. The output from such a set in such a plane makes up a feature map, as seen in figure 1. Units in one feature map now have the constraint to perform the same operation on different parts of the image. Usually a convolutional layer is composed of several feature maps. This way multiple features can be extracted at each location. For the first hidden layer in figure 1 it consists of four feature maps with 5x5 receptive fields[2].

If the input for a convolutional layer would get shiftings it would mean that the output will also be shifted, but it will leave it unchanged otherwise. When a feature has been detected its exact location is not too important, this is if the features approximate position relative to other features is preserved. Because of this each convolutional layer in LeCun is followed by an additional layer which performs a local averaging and a subsampling to reduce the resolution of the feature map and reducing the sensitivity of the output to shifts and distortions[2]. This is being done in the second and third hidden layer if figure 1.

It is often of common practice to alternate layers of convolutions and subsampling. At each layer the number of feature maps is increased as when the spatial resolution is decreased.

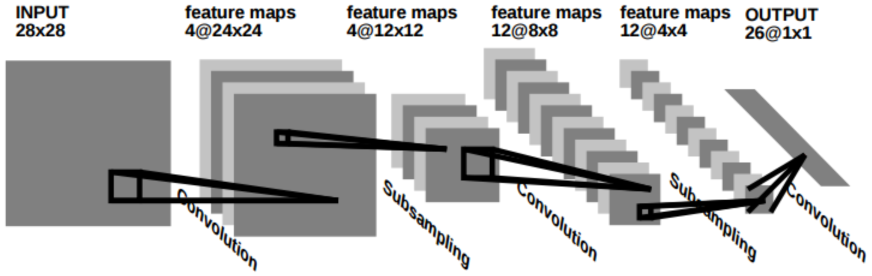


Fig. 1: LeNet-5 CNN for handwritten characters recognition.

2.2 Data augmentation

One technique to try and overcome or diminish the effect of overfitting is data augmentation. Data augmentation is the technique of artificially enlarge ones data set using label-preserving transformations[4]. This by performing one or more deformations to training samples which result in new, additional training data[5].

An important aspect of augmentation is that in order for it to be valid the deformations applied to the labeled data do not change the semantic meaning of the labels[5]. That is if for example after deforming an image of a airplane it should still be an image of a airplaine.

Next we will list all the different augmentation techniques used in the networks built in section 3.2.

90 degrees rotations Randomly rotate a image 90 degrees zero, one, two or three times.

Blur Randomly blur an imabge by apllying a gaussian filter with a random sigma. Sigma between 0 and 0.5.

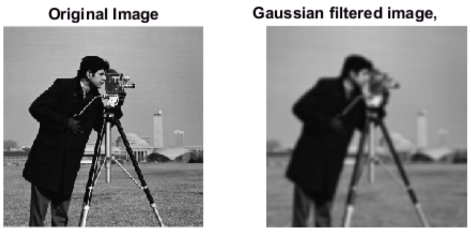
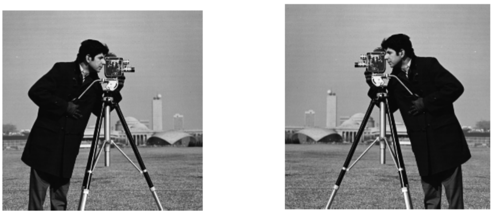


Fig. 2: Example of image blurring

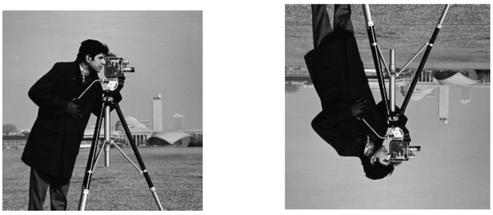
Flip left to right Horizontally flip an image.



(a) Original (b) After flip

Fig. 3: Example of flipping image horizontally

Flip upside down Vertically flip an image



(a) Original (b) After flip

Fig. 4: Example of flipping image vertically

Rotation Rotate an image x degrees. For our networks we will randomly rotate images in the span of -20 degrees to 20 degrees.

3 Approach

To be able to make comparisons of the the effect different augmentation techniques have an on a network, the same network would have to be made and train multiple times where only the augmentation techniques used would differ.

For this reason a base network where created that was to be used by all the networks.

This network consists of three convolutional layers. Maxpooling between the first and second convolutional layer and after the third convolutional layer. And lastly two fully connected layers with dropout between the two. The network is ilustrated in figure 5

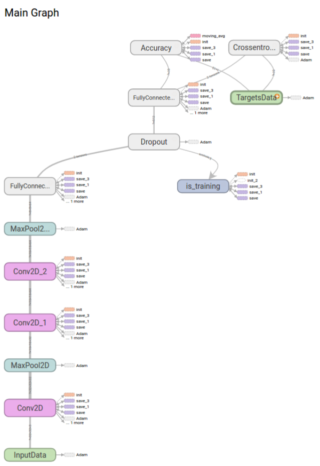


Fig. 5: Deep convolutional neural network for training

3.1 Method

For implementation purposes Tensorflow[6], developed by Google, was used. Tensorflow is an open source software library for numerical computation that uses data flow graphs. It was developed for the purposes of conducting machine learning and deep neural networks research. Alongside tensorflow CUDA Toolkit[6] and cuDNN[7] were installed for GPU support.

Data The data used for the network was the CIFAR-10[1] dataset. CIFAR-10 consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. With 50000 images as a training set and 10000 images as a test set.

3.2 Experiments

The following augmentation reprisentation where used:

– Random 90degrees rotation. Id 1.
– Random blur. Id 2.

- Random flip left to right. Id 3.
- Random flip up and down. Id 4.
- Random rotation, with max angle set to 25 degrees. Id 5.

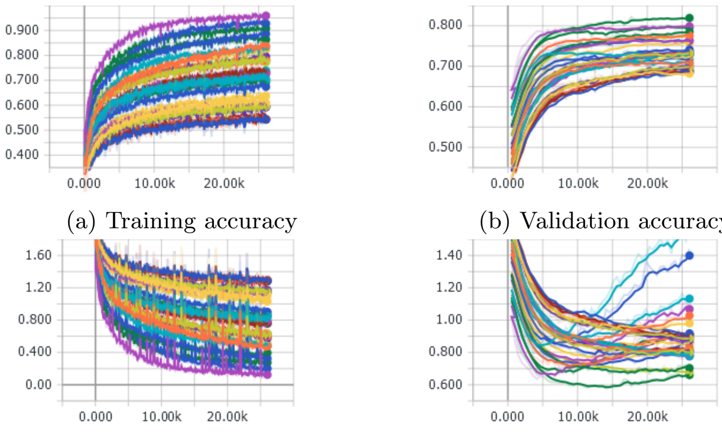
For each unique combination of augmentations a network, the one described in section 3, where trained and data on training and validation accuracy as well as loss where retrieved. Each network where trained for 50 epochs and with a batch size of 96 samples.

The following networks where created (with the numbers representing which augmentation techniques where used):

- (), the one without augmentation.
- (1),(2),(3),(4),(5)
- (1, 2),(1, 3),(1, 4),(1, 5),(2, 3),(2, 4),(2, 5),(3, 4),(3, 5), (4, 5)
- (1, 2, 3),(1, 2, 4),(1, 2, 5),(1, 3, 4),(1, 3, 5),(1, 4, 5), (2, 3, 4),(2, 3, 5), (2, 4, 5),(3, 4, 5)
- (1, 2, 3, 4),(1, 2, 3, 5),(1, 2, 4, 5),(1, 3, 4, 5),(2, 3, 4, 5)
- (1, 2, 3, 4, 5)

4 Results

The following figure 6 shows the results from all 32 networks. The result is messured with training- and validation accuracy aswell as training- and validation loss.

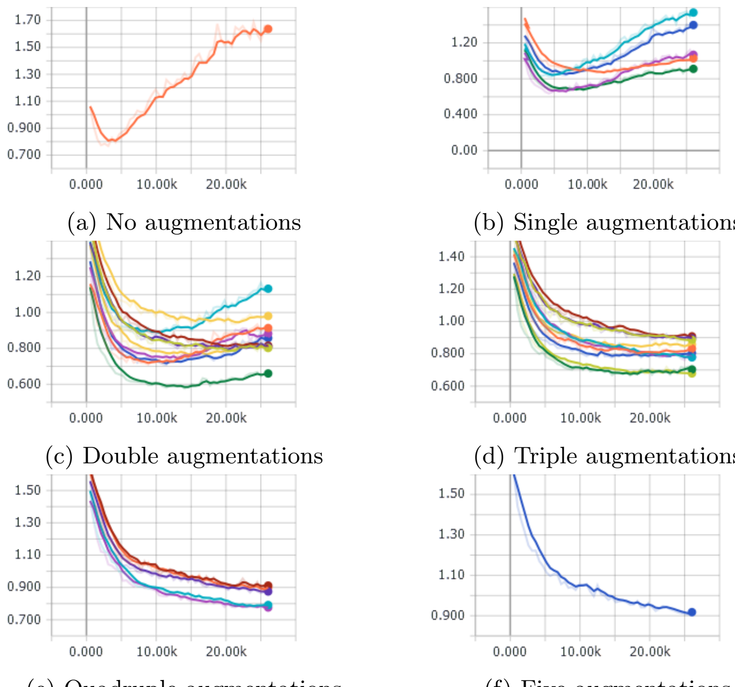


(a) Training accuracy (b) Validation accuracy

(c) Training loss (d) Validation loss

Fig. 6: All networks

Figure 7 shows the result of all the different networks sorted by the different types of groups. The ressult messured with the validation loss.



(a) No augmentations (b) Single augmentations (c) Double augmentations (d) Triple augmentations (e) Quadruple augmentations (f) Five augmentations

Fig. 7: All networks

The best performing network achived a test accurace of ≈96% and a validation accuracy of ≈82% and used the augmentations of random flip left to right and random rotation.

5 Discussion and conclusions

With performance evaluated by networks accuracy on the validation set, the network with the two augmentation techniques implemented flip left to right and rotation performed the best.

Now lets consider the performance evaluated by how much the networks are overfitting. First we can note that the network greatly overfitts without any augmentation, see figure 7a. Looking at figure 6d it becomes clear that augmentation have an important effect on overfitting. Some networks loss starts to increase really fast after decreasing in the beginning, i.e. overfitting the data.

Lets consider the sub-figures in figure7. As mention earlier without augmentation, 7a, greatly overfitts early, only managing to come down to 0.8 loss. After introducing only one augmentation technique it vastly improves the performance, 7b, though there are differences between techniques. The blue network (which is bluring) overfitts more then the green network (which is rotation). The blue network validation scores ≈73% as opposed to the green that scores ≈79%.

As for the double-augmentation networks, figure 7c, we get overall smother curves with lesser overfitting and for the triple-augmentation, figure 7d, the curves becomes even smother with no network really overfitting anymore. Though the networks with lowest loss in double- and triple-augmentation networks ends around 0.7.

After having three times augmentation adding more seams to drasticly slow down training. In figure 7d the best network only manage to get down to a loss of ≈0.8 and in figure 7e the network does not even reach bellow 0.9. Further studies would be required with longer training time to see how these networks evolve. Do they level out at around 0.70 as the other networks or are they capable of reaching even lower?

References

1. : Cifar10 the cifar-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html
2. LeCun, Y., Bengio, Y.: The handbook of brain theory and neural networks. MIT Press, Cambridge, MA, USA (1998) 255–258
3. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (Nov 1998) 2278–2324
4. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 25. Curran Associates, Inc. (2012) 1097–1105
5. Salamon, J., Bello, J.P.: Deep convolutional neural networks and data augmentation for environmental sound classification. CoRR **abs/1608.04363** (2016)
6. : CUDA Toolkit the cifar-10 dataset. http://docs.nvidia.com/cuda/index.html\#axzz4qlpHwFpl
7. : NVIDIA cuDNN the cifar-10 dataset. https://developer.nvidia.com/cudnn
8. Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: High-performance neural networks for visual object classification. CoRR **abs/1102.0183** (2011)
9. : Tensorflow the cifar-10 dataset. https://www.tensorflow.org/