Assignment-1 of Deep Learning in Computer Vision

Xiao Hu[1], Boya Wu[2], and Wenjing Zhao[3]

1. xiahaa@space.dtu.dk, 2. s170061@student.dtu.dk, 3. s191118@student.dtu.dk

1 Image Classification: Hotdog/not hotdog

We have investigated a two-classes image classification problem by dividing images into two categories: Hotdog and Not-hotdog. Data got from ImageNet[1] has been provided and reassigned into two classes by the instructor. The following tasks have been done:

- Design CNN and retrain pretrained model to solve the classification problem.
- Improve generalization of CNN by data augmentation, input normalization, early stopping.
- Optimize CNN with different optimizers and evaluate their performance.

We realize that there are other factors that we have not applied, but may also improve the performance of the CNN, such as initialization[2], weight decaying, etc.

1.1 Network Design

We start with self-designed CNN. The first CNN we design is inspired from VGG [2], as shown in Figure 6, which includes 5 convolutional layers and 2 fully connected layers. We design each convolutional layer as follows: *Conv2d, ReLU, Conv2d, ReLU, MaxPool2d*. For convolution, we keep using kernel size (3×3) with stride being 1 and padding being 1.

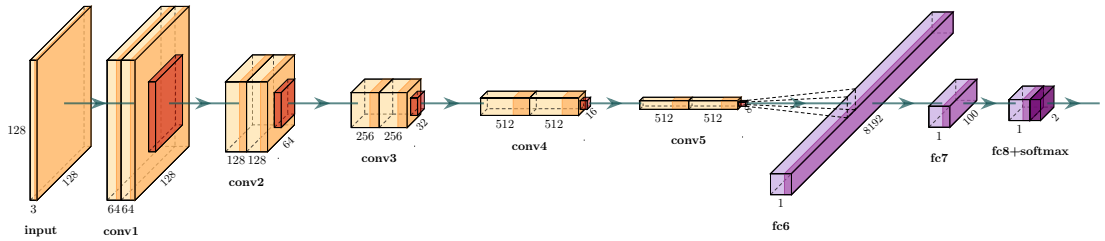

Fig. 1. Self-designed CNN.

The second model we used is the well-known VGG-16 [2], provided by PYTORCH as a built-in pre-trained network model. Since VGG-16 is trained on the ImageNet challenge, in order to use it for out task, we have to retain it. Considering the large number of parameters of VGG-16, we decide to freeze the feature layer and the average pooling layer of VGG-16, but retrain the classifier layer on our dataset. Since the struture of VGG-16 is well known, we omit it for brevity.

1.2 Generalization

Data Augmentation We use data augmentation to create more data by exploiting things that the label is invariant towards. Training data has been augmented by the following chain:

- Resize: the raw image is resized to 256×256.
- RandomResizedCrop: randomly crop 90% portion from the resized image and output as a 224×224 image.
- RandomHorizontalFlip: Horizontally flip randomly ($p = 0.5$).
- RandomRotation: $+/-$ 45 degree.

[1] http://www.image-net.org/

[2] Default weight initialization [1] is used.

Input Normalization Besides the augmentation part, we use input data normalization to normalize the input to zero mean and standard deviation of 1.

Fig. 2. Left image shows the results after data augmentation and right image shows the results after input normalization.

An example of a mini batch after data augmentation and input normalization is shown in Figure 2. According to our experiments, the test accuracy could increase by 2 to 3 percents thanks to data augmentation and input normalization.

1.3 Optimization

Due to the highly nonlinearity, the neural network is a highly non-convex function with large number of local minimums. Training with different optimizers and different learning rates may converge to different local minimums. Therefore, we also evaluate the performance of the trained network with optimization options.

Optimiser Four types of optimiser have been tried: **SGD, SGD with momentum, RMSProp and Adam**. The result of applying these optimisers[3] is given in Table 1. As we can see, when learning rate is 0.0001 and the number epoch equals to 15, Adam optimiser provides higher accuracy on both training set and test set than other three types. SGD optimiser and SGD with momentum optimiser could not achieve convergence effectively within 15 epochs with the learning rate being $1e^{-4}$. They need to spend twice as much time as Adam optimiser. Therefore, we decided to use Adam as our optimiser in this project.

Optimizer Type	Train Accuracy	Test Accuracy	Spending Time	Epochs at convergence
SGD	88.4%	90.9%	4:31	30
SGD with Momentum	92.3%	93.1%	4:36	25
Adam	96.9%	93.9%	4:54	15
RMSProp	95.8%	93.1%	4:49	15

Table 1. Optimizer Analysis: $lr = 1e-4$ and $epoch = 15$

Learning Rate We evaluate the training performance with different learning rates, as shown in in Table 2 and Figure 3. As we could see in Figure 3, both training data set and testing data set get the highest accuracy when learning rate=$4e^{-5}$. Our experimental results suggest the learning rates equals to $4e-5$.

1.4 Result & Analysis

Since the performance of VGG-16 is much better than our self-designed model, we finally select VGG-16 as the model for image classification. After retrained VGG-16 with our selected generalization and optimization, the accuracy on the test dataset is about 94.2%, see Figure 4.

[3] We kept other settings unchanged but only switch different optimisers.

lr	Train Acc.	Test Acc.	Spending Time
5e-6	94.7%	93.4%	4:54
1e-5	95.7%	93.4%	4:54
2e-5	96.8%	93.6%	4:54
3e-5	97.2%	93.6%	4:55
4e-5	97.5%	94.2%	4:56
5e-5	97.4%	93.3%	4:54
1e-4	97.0%	92.4%	4:54
5e-4	95.4%	93.6%	4:56
1e-3	94.2%	92.3%	4:56

Table 2. Learning Rate Analysis: *epoch* = 15.

Fig. 3. Evaluation of accuracy versus to learning rate.

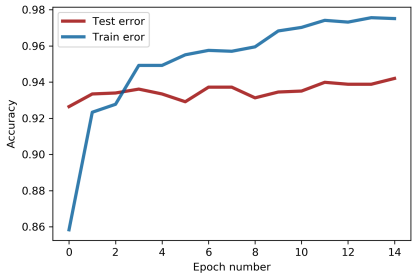

Fig. 4. Accuracy Curve of VGG-16 retrained with Adam and learning rate = 4e-5.

We analyzed false classified images and then divided them into three groups, see Figure 5:

1. The two main components of a hot dog: the bread and sausage are placed separately in the image which is different from the majority of hot dogs' shapes in the training set.
2. Hot dogs only occupy a minority part of the picture. On the other hand, human hand or body occupy the majority part. This may cause it is hard to notice the tiny hot dog in the image.
3. Hot dogs are occluded by vegetables, cheese or other things.

2 Object Detection: Where the party at?

We have investigated a object detection problem: to detect the digital numbers inside an image. The Street View House Numbers (SVHN) dataset[4] are used for this task. The following tasks have been done:

- Design and train a CNN that can classify a patch as digits or not.
- Apply the convolutional implementation of sliding window to detect interest objects within the full size image.
- Use non-maximum suppression and the Intersection over Union (IoU) to select the strongest detection.

2.1 Data Preparation

Since SVHN only contains digital images, we also need non-digital images. We have tried two options, see Figure 7 as an example:

- CIFAR10[5] contains images of the following classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.
- Cropped images from full size images of SVHN dataset which have no overlaps with digital bounding boxes: 13375 test images and 26888 train images.

Fig. 5. False classified images: group 1 (left), group 2 (middle), and group 3 (right).

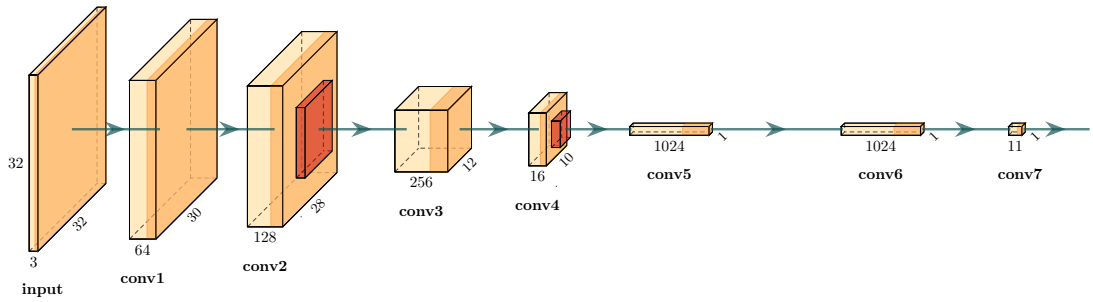

Fig. 6. CNN model.

The intuitive may be that the two options are the same since they are just using as the non-digits class. However, after experiment, we realized that the second option is much better than the first option. By visualizing the full size images, we can find that the probability of seeing digits and objects of the ten classes of CIFAR10 is quite low. Therefore, if we use CIFAR10 as the non-digits class, then the network could classify objects belonging to the ten classes of CIFAR10 as non digits. However, the network may not have the capability of classifying a dirty wall or an ridge as non digits. Consequently, we finally decide to use the second option at the end.

2.2 Network Design

To meet the requirement of the convolutional implementation of sliding window, we need to transform all fully connected layers to equivalent convolutional layers. The CNN we design is shown in Figure 6, which includes 7 convolutional layers. We add two max-pooling layers. For convolution, we keep using kernel size (3×3) with stride being 1 and no padding.

[4] http://udl.stanford.edu/housenumbers/
[5] https://www.cs.toronto.edu/ kriz/cifar.html

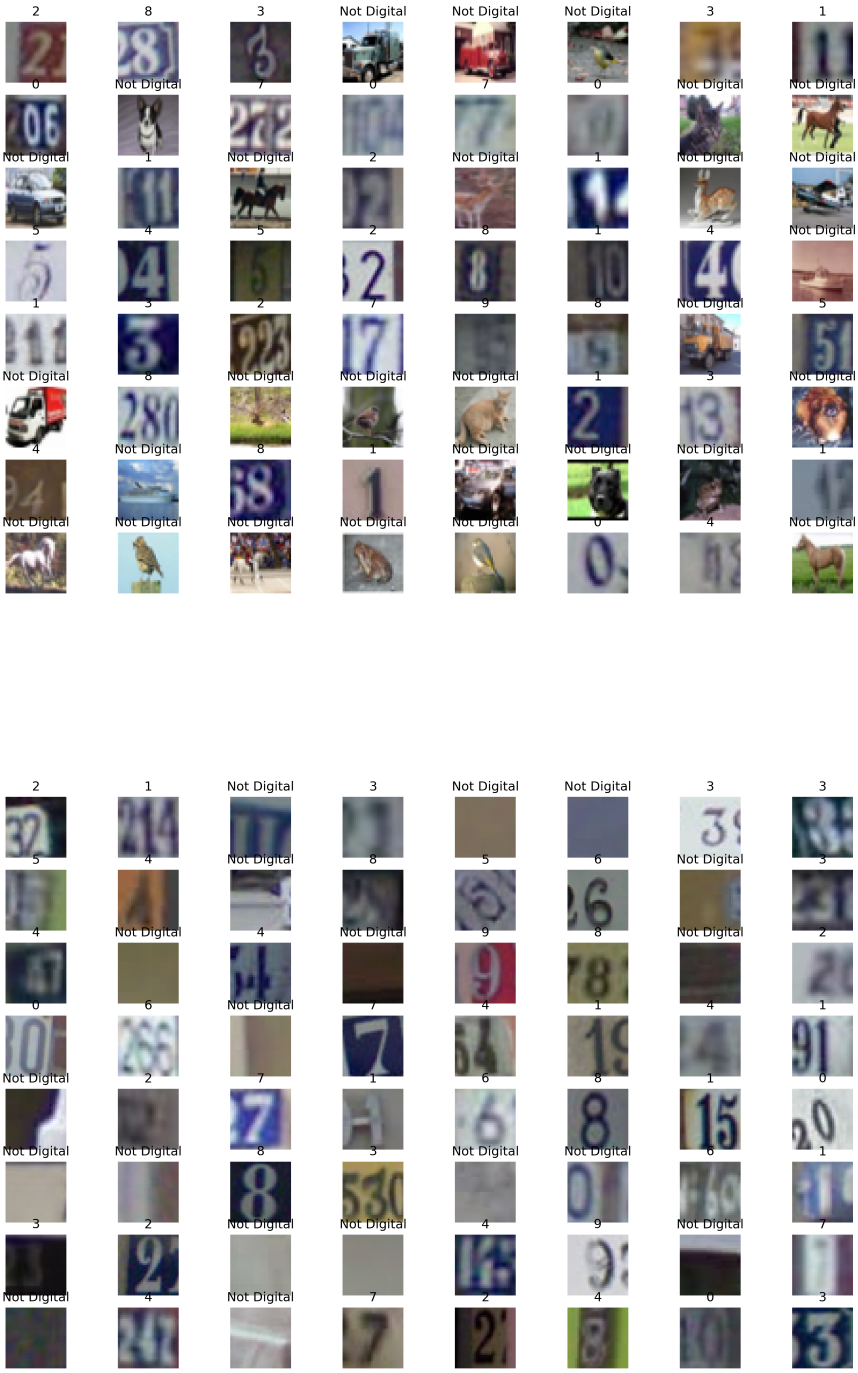

Fig. 7. Left image shows the dataset using CIFAR10 as non digits and right image shows the dataset using cropped images as non digits.

2.3 Generalization & Optimization

We use input normalization to normalize the input to zero mean and standard deviation of 1.

For optimization, we use Adam as the optimizer with learning rate being $1e^{-3}$ and run training for 20 epochs. The final training and test accuracy are 98.1% and 95.1%, respectively.

2.4 Convolutional implementation of the sliding win- dow algorithm

Since we have designed the network by using convolutional layers as a replacements of fully connected layers, we can directly apply the network to the full size image (**only apply forward without final** $softmax$). Considering we are using two max pooling layers, so this convolutional implementation is equivalent of running the sliding window (32×32) with a stride of 4×4.

When we obtain the convolution response from the network, we can then classify each grids as digits or not. For all digital grids, we map them back to their corresponding sliding windows.

2.5 Non-Maximum Suppression & IoU

We apply the non-maximum suppression as follows:

1. Remove detections with probability less than τ_1;
2. For each class: start with the bounding box with the highest probability, find its overlapped bounding boxes, compute IoU and remove bounding boxes with an IoU over tau_2;

We finally plot all detections as bounding boxes with the titles being the corresponding labels.

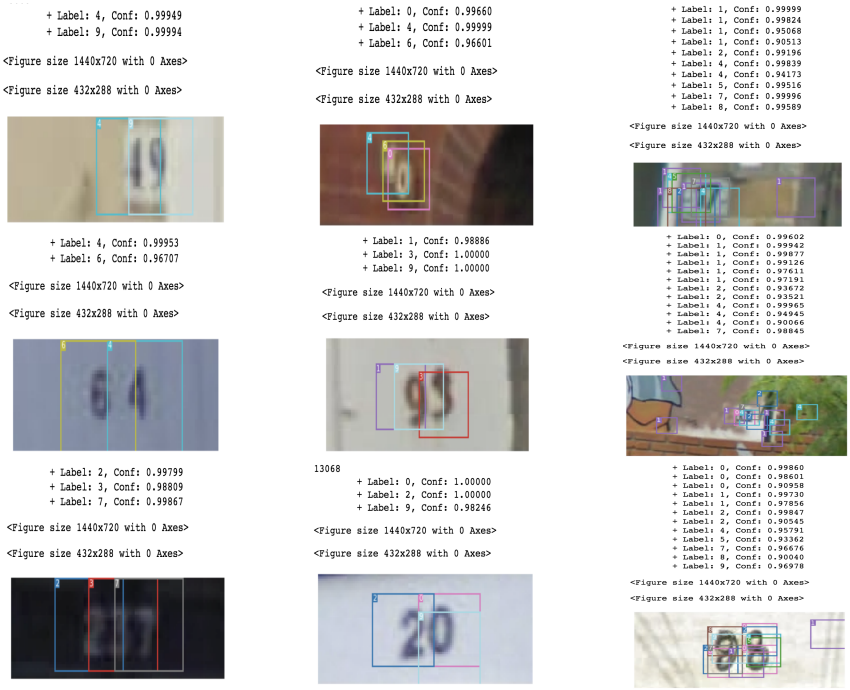

Fig. 8. Detection results: Good (left), Normal (middle), and Poor (right).

2.6 Results & Analysis

The final detection results are shown in Figure 8. We divided the results into three categories:

- Good: all correct detections.
- Normal: more correct detections than false positives.

– Poor: many false positives.

Those results are obtained by using the cropped images as the non-digits class. Results obtained using CIFAR10 are much worse, so they are omitted for brevity.

From those results, it can be seen that the implemented detection algorithm is able to detect digits in some real images. However, several points need to be considered for improving its performance:

– Generate more training images using cropped images: we are using a stride of 96×96 to slide a window in the full size image and discard any patches with any overlaps with the bounding boxes. This could be conservative that patches with partial digits will be thrown. If we can use more data (we didn't due to the time constraint since copying large data to server and process is really time-consuming), then the performance may improve.
– Data augmentation: since the SVHN train images are cropped and resized versions of the raw image, the performance of the trained network may drop a little bit on the raw image. Considering data augmentation will help the network to recognize digits in raw image.
– Train a network that not only can classify digits/not-digits, but also can propose the location of the bounding box. This may significantly improve the performance. As it can be seen from those results, another weakness of this algorithm is that the location of the proposed bounding box is not very accurate.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)