Danmarks Tekniske Universitet

DTU

02514 Deep learning in computer vision Report 1

Authors

Elika Araghi - s181740
Grímur Zimsen - s191685
Ghassen Lassoued - s196609

June 26, 2020

Contents

1 Introduction

The goal of this project was to gain a deeper understanding of convolutional neural networks and involved classifying correctly hotdogs and locating house numbers.

2 Hotdog/not hotdog classification

The first part of the project consists of the classification of images coming from ImageNet using convolutional neural network. The task is to identify whether the image contains a hotdog or not. The images all have different resolutions and aspect ratios. It is essential that they have the same resolution to be implemented in the forward pass. Thus all images are resized to 128x128.

2.1 Architecture

In order to overcome overfitting and get better test accuracy, different approaches are combined. The idea is to fit as many models as possible with the different techniques described below. Around 63 convolutional neural networks with different combinations of techniques were implemented.

We started with a very simple network. Then we improve the network by increasing the complexity of the network by adding more layers and increasing the features' size. Then, to avoid overfitting, we vary one parameter or technique described below while the others are fixed, find the best model, change the following parameter or technique, find the best model etc.

2.1.1 Optimization

Three choices were compared. We implemented different models with optimizers: Stochastic gradient descent (SGD), Adam or AdamW.

2.1.2 Regularization

Dropout

Neural networks have many weights and can easily overfit. A solution to this is dropout layers which were introduced with different probabilities of an element to be zeroed (0.1/ 0.2/ 0.5). Thus this is an efficient way of performing model averaging with neural networks. These layers were introduced in both convolution part and fully connected part of the network.

Weight decay

Different weight decays (L2 penalty) ranging from 0.001 to 0.2 were implemented. The way they work is that the weight decay is multiplied with the sum of squares of the parameters and added to the loss function. This penalizes complexity.

Batch normalization

Batch normalization was implemented to a various degree, both on convolutional and fully connected layers and also on just one or the other. The purpose of batch normalization is to increase stability and improve convergence.

2.1.3 Data augmentation

The images were augmented by randomly rotating the images by ±5 degrees, translating them up to 5% on each axis, scaling them up and down by 5% and shearing them by ±5 degrees. The purpose of this is to enrich the training data with more varied images.

2.1.4 Learning rate, batch size, number of epochs, activation function

Different choices were made:

- Learning rate: 0.1/ 0.001/ 0.0001/ 0.00001
- Batch size: 64/ 128
- Number of epochs: 5/ 10/ 20/ 30
- Activation function: ReLU/ Leaky ReLU

2.1.5 Network accuracy and misclassification

In order to try and come up with intuitive reasons for the misclassified images, below are 8 images of correct classification and 8 images of wrong classification.

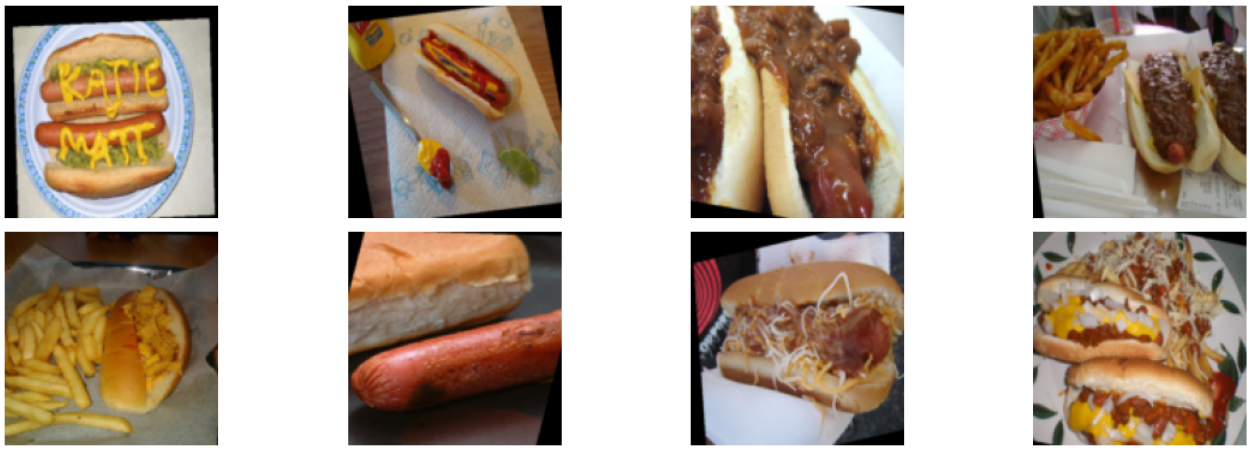


Figure 1: Correct classification

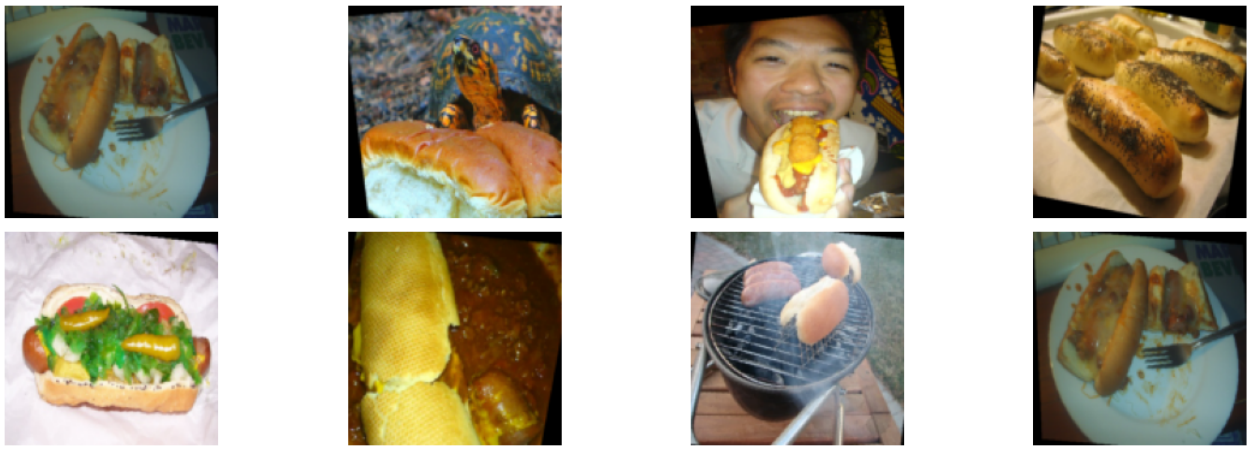


Figure 2: Wrong classification

One can remark that in the correctly classified images there is often one hotdog, and that the image is zoomed on the hotdog.
One can also remark that the misclassified images contain many hotdogs, humans, other objects etc.
Note that one can not be sure of the reasons of misclassification and can not naively say that the reasons above are behind the misclassification.
The adversarial example below explains this.

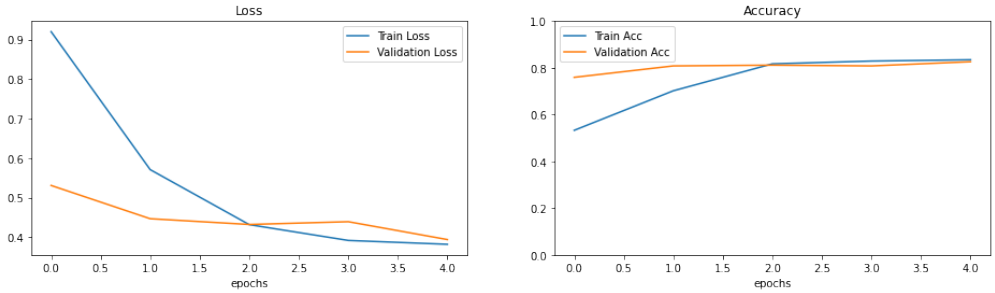
2.1.6 Adversarial example

By creating an image of random noise and using the model to iterate the pixel values, it is possible to increase the probability of the image being classified as a hotdog to almost 100%. The resulting image is still just random noise. This is because CNNs do not perceive images in the same way humans do since they are merely matrix multiplications with some added non-linearities.
For the example computed, the probability of being classified as hotdog went from 6.9702e-30 to 0.9970

2.1.7 Transfer learning

Transfer learning is also implemented using built-in model VGG16 in Keras. The pre-trained weights are used for the convolutional layers. Though we fine-tuned the fully connected layers and changed the number of outputs of the last layer from 1000 to 2 (as we have 2 classes: Hot dog and not a hot dog). The network contained 134,260,544 non-trainable parameters and 8,194 trainable parameters.

Accuracy and Loss for train and validation sets are shown in the figure below. The network reaches around 80 percent accuracy only after two epochs.



Train Loss	Accuracy	Validation Loss	Validation Accuracy
0.9199	0.5328	0.5311	0.7593
0.5711	0.7018	0.4469	0.8081
0.4319	0.8170	0.4324	0.8114
0.3920	0.8289	0.4393	0.8081
0.3823	0.8345	0.3942	0.8260

The final chosen model is obtained by transfer learning of the built-in model VGG16 because it gives the best results.

3 Street View House Numbering - Object detection

The second part of the project consists of first training a convolutional neural network to classfiy digits from 0 to 9. The model is trained using Street View House Numbers (SVHN) dataset which contains 32x32 images of house numbers.

While training the network, it is very important to use images that do no contain digits to avoid false positives. This is done by creating the 11th class for negative samples meaning 'no digit' using CIFAR10 dataset.

Then object detection is performed using the classification network as 32x32 sliding window of larger SVHN images.

3.1 Data Loader

The built-in dataset from pytorch is used to train the network and all images are of the size 32x32.

3.2 Architecture

We started with a very simple network. Then we improve the network by repeating these two steps:

- Increased the complexity of the network by adding more layers and increasing the features' size.
- Tune regularization (dropout, batch-normalization, weight decay) to avoid over fitting.

Our final structure is as follow:

- Conv3/20 (5x5)
- Dropout(0.2)
- Maxpool (2x2)
- BatchNorm
- Conv20/30 (3x3)
- Dropout(0.2)
- Maxpool (2x2)
- BatchNorm
- Conv30/300 (6x6)
- Conv300/11

3.3 Localization

Note: Unfortunately, it was not possible to completely finish the implementation due to lack of time. The missing functionality will be explained as best as possible.

Once the neural network has been trained using the 32x32 images, it is possible to run through it larger images to obtain a tensor with 11 channels. The first 10 channels represent the house numbers ranging from 0 to 9. The values in the matrices in each channel represent the probabilities of corresponding house numbers being present in the original image. The coordinates of said values within the matrices reveal where the house numbers are located in the image. Only values of 0.6 or higher are taken into consideration as possible house numbers. This is referred to as non-maximum suppression.

By backtracking through the architecture, it is possible to both obtain the location of the object in the original image and the bounding box. In order to obtain bounding boxes of different sizes, the original image is simply scaled up or down and run through the network again. This results in numerous overlapping boxes which are then cleaned up using the Jaccard index. That involves dividing the area of overlap with the area of union of bounding boxes of the same class to obtain an IoU (intersection over union) coefficient. Any boxes with IoU higher than 0.5 are discarded.

4 Conclusion

In the first part of the project it is shown how to perform a binary classification using convolutional neural network. The results depend on its architecture. This task shows how to get a compromise by increasing complexity and the neural network and then by using the techniques described to avoid overfitting.

The group's efforts in the second part reached as far as finding the center of the house numbers in the original image. The correct bounding boxes remained elusive, possibly due to undiscovered errors in the code. Despite these shortcomings, the project provided valuable insight into the inner workings of convolutional neural networks and object localization.

5 Work process

All method used were discussed between group members.
Monday 8.6

- Elika: Fine-tuning CNN
- Ghassen: Fine-tuning CNN
- Grímur: Fine-tuning CNN

Tuesday 9.6

- Elika: Fine-tuning CNN, VGG
- Ghassen: Fine-tuning CNN, report writing
- Grímur: Fine-tuning CNN, Adversarial example

Wednesday 10.6

- Elika: Data preparation
- Ghassen: Data preparation
- Grímur: Data preparation

Thursday 11.6

- Elika: Fine-tuning CNN, backtracking of network, report writing
- Ghassen: Fine-tuning CNN, report writing
- Grímur: Fine-tuning CNN, localization, report writing

List of Figures