Vehicle Classification Using Convolutional Neural Networks and Fast Fourier Transform

Viktoria Plemakova
University of Tartu
Institute of Computer Science
Email: viktoria.plemakova@ut.ee

***Abstract*—Recently convolutional neural networks have gained popularity in the field of computer vision. Great results have been achieved when using convolutional neural networks to detect and classify various objects in images, e.g. cats, dogs, cars etc. In this work such neural network technique is used to tell vehicles from non-vehicles. Furthermore, we investigate whether applying Discrete Fourier Transform to images during preprocessing step improves the overall accuracy of the network.**

I. Introduction

Vehicle detection, tracking and classification in either images or videos is an importand task in the field of intelligent transportation systems. Convolutional neural networks (CNN) have become a popular solution to this kind of problem. Detecting and extracting features from image data is the strong point of CNNs. An advantage of CNNs is that they require little to no preprocessing at all to achieve good results [1].

In this paper CNN is applied to a binary classification task, namely the network should be able to distinguish between vehicles (rear-view images) and non-vehicles (random road parts). In addition to that, Fast Fourier Transform (FFT) is used in the preprocessing stage to test whether it affects the accuracy of the network.

The rest of this paper is organized as follows. Section II introduces convolutional neural networks. Section III introduces the implementation. In that section the dataset, preprocessing and developed network architecture is described. Section IV presents the results of the experiments. Conclusions are made in Section V.

II. Background

A convolutional neural network (CNN) is a deep learning technique. The idea of such network was proposed by LeCun *et al.* [2] in 1990. The neural network described in their paper performed handwritten digit recognition. Later, LeCun *et al.* [1] published the architecture of LeNet-5, the network that the authors used to perform their experiments. Over the years, CNNs have proved to perform especially well on image data.

Since CNNs are a variation of deep neural networks, they share some similar elements. For example, CNNs have an input and output layer as well as one or more hidden layers in between. Three main components of CNNs are convolutional, pooling and fully-connected layers. Each layer contains computational units similarly to deep neural networks. A common CNN architecture is illustrated in Figure 1.

A. Convolutional layer

A convolutional layer is the main part of CNNs. It performs feature learning and extraction. Each convolutional layer outputs multiple feature maps that are the result of applying several filters of size $n \times n$ to every input. Input images are simply matrices of pixel values of size $n \times m \times 3$ in case of color images and $n \times m \times 1$ in case of grayscale input. Element-wise multiplication is performed between the kernel and every input region. The products are summed up into a single value.

B. Pooling layer

A pooling layer is usually used after convolutional layers. Feature maps serve as input to pooling layers. We are not interested in the exact position of every detected feature [1]. The pooling layer helps to leave only the most useful information. There are different types of operations that can be performed in this layer. The most popular choices are max pooling and average pooling. In case of max pooling, the largest value of each input region is selected as illustrated in Figure 2.

C. Fully-connected layer

After we do not want to add any more convolutional or pooling layers, everything is flattened into a vector. This is succeeded by one or multiple fully-connected layers. The number of computational units in the final layer depends on a task. For example, if there were 10 classes in a classification task, there would be 10 units in the final layer. In case of binary classification, there is one output unit.

D. Activation functions

An activation function f is applied between the hidden layers to every layer output. In case of CNNs the most popular activation function is Rectified Linear Unit (ReLU)

$$f(x) = max(0, x) \tag{1}$$

Additionally to ReLU, other frequently used activation functions include sigmoid

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

and hyperbolic tangent

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{3}$$

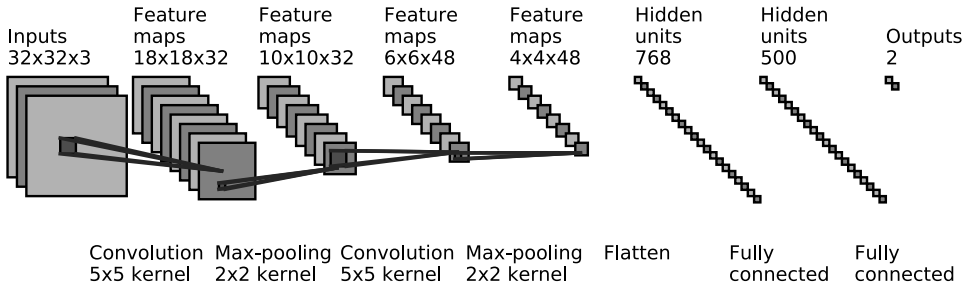

Fig. 1. A typical CNN architecture.

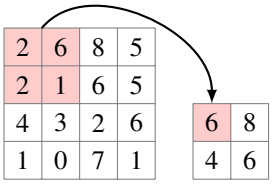

Fig. 2. Example of 2×2 max pooling.

Fig. 3. A sample of vehicle and background images in the dataset.

III. Methods

This section introduces the methodology of the experiments. Firstly, this section described the dataset that is used for training and testing the CNN. Next, the preprocessing steps and the architecture of the network are presented. The final solution uses Keras [3], a Python library for neural networks, with Tensorflow backend.

A. Dataset description

The GTI-UPM Vehicle Image Database [4] was used to train and test the network. It includes images that were extracted from video sequences recorded in Madrid, Brussels and Turin. The dataset consists of about 7000 images that are divided into 2 classes, vehicles and non-vehicles, as shown in Figure 3. The images are of size 64×64 and selected in a way that some of them contain vehicles only partially. Additionally, the images reflect different weather and lighting conditions, e.g. some images were taken during a cloudy day or in the evening when there is less light.

B. Preprocessing

To start with, the dataset images are converted to grayscale. Then, the dataset is normalized using standardization technique. Standardization transforms the data in a way that its mean is 0 and the standard deviation is equal to 1. It is defined by:

$$Z = \frac{X - \mu}{\sigma}$$

where μ is the mean and σ the standard deviation.

In addition to that, the `ImageDataGenerator` from Keras is used to generate more training data from the existing dataset. Data augmentation can improve the final accuracy of the model since it will learn from a wider variety of examples. Additionally, this technique can reduce overfitting when using small datasets [5]. Data augmentation usually involves operations such as rotating, flipping or shifting images. For instance, in this work the images are randomly rotated, shifted and zoomed as well as vertically and horizontally flipped, as can be seen in Figure 4. Data augmentation was used only on the training set.

One of the goals is to investigate how the network behaves when the images are preprocessed using Fast Fourier Transform (FFT). This is an algorithm that speeds up the computation of Discrete Fourier Transform (DFT). DFT of an image of size $M \times N$ is defined by [6]:

$$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \exp\left[-i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)\right]$$

where $F(u,v)$ is the transformed image in the frequency domain and $f(m,n)$ is the original image in the spatial domain. To better visualize an image in the frequency domain we first shift the $F(0,0)$ value to the center of the image. The

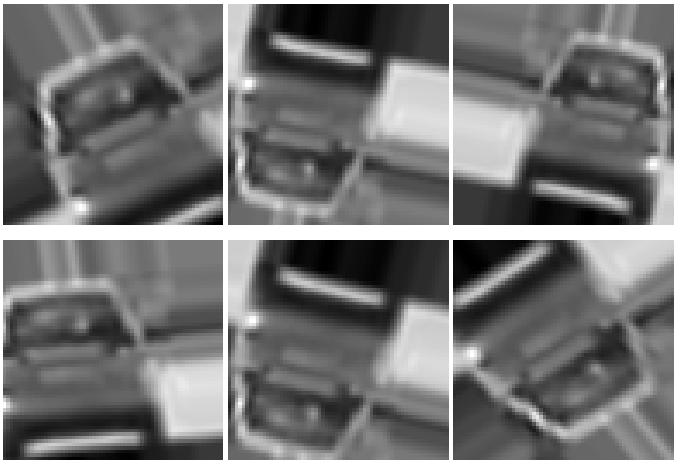

Fig. 4. A sample of data augmentation. A vehicle image is transformed in various ways.

higher frequencies are positioned away from the center. The second step is to take the logarithm [6]:

$$Z(u, v) = log(|F(u, v)|).$$

An example of described procedure is illustrated in Figure 5.

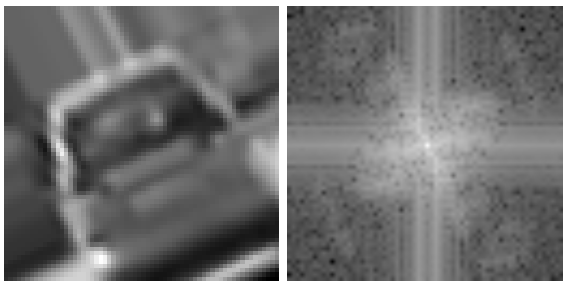

Fig. 5. A vehicle image and the same image after applying FFT.

C. Network architecture

The CNN, used in the experiments, contains 6 convolutional layers, 3 pooling layers and 1 fully-connected layer. Every two consecutive convolutional layers followed by a sub-sampling layer. This is slightly similar to how the convolutional layers are stacked in VGGNet [7] architecture. The input expects images of size 64×64×1. The ReLU function is used as the activation function between the layers. This is the description of the final architecture (see Figure 6) after trying out more simpler models. Less complicated architectures, i.e. fewer layers and kernels, tended to produce an overfitting problem.

The first two convolutional layers have 32 filters of size 3×3. They are followed by a sub-sampling layer that uses the max pooling operation with the size of 2×2. The second stack of convolutional layers has 64 filters with a size of 3×3. Similarly to the first convolutional layers, they are followed by a max-pooling layer with the kernel size of 2×2. The final two convolutional layers have 128 filters of size 3×3 and they

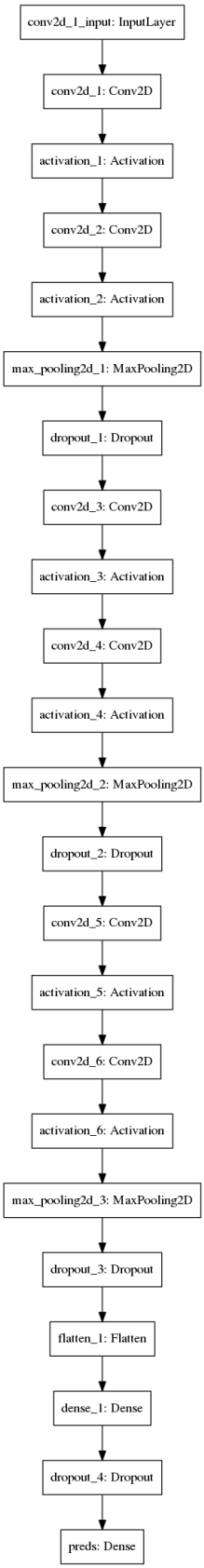

Fig. 6. The proposed network architecture.

are followed by the identical sub-sampling layer like previous convolutional layers.

The fully-connected layer has 128 units and dropout is applied. During the training dropout is applied after every sub-sampling layer as well. Finally, the output layer has one unit. A sigmoid activation is used in the final layer since it is a binary classification problem.

IV. Results

The network was trained and tested on the Rocket cluster of the University of Tartu to speed up the process. The dataset was split into the training and testing sets where the training set included 5860 images and testing was done on 1465 images. For every experiment the model was trained for 10 epochs and Adam optimizer was used with the learning rate of 0.001.

A. Evaluation metrics

A confusion matrix can be used to evaluate the results of the classification. As illustrated in Table I the columns of the matrix belong to the predicted classes and the rows to the actual labels. True positive (TP) and true negative (TN) cells show the amount of correctly estimated positive and negative examples. On the contrary, false positive (FP) and false negative (FN) represent incorrect estimations of either positive or negative examples.

TABLE I
CONFUSION MATRIX.

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

B. Results

Firstly, the network was trained without using FFT during the preprocessing step. After 10 epochs the training accuracy was already 95.70% and the final test accuracy 95.97%. The confusion matrix of this experiment is provided in Figure 7. Based on the confusion matrix it can be concluded that the network was able to perform the classification task well. There were only 59 misclassifications in total.

The second model was identical to the previous one but additionally, FFT was used to preprocess images. After the same amount of epochs the training accuracy was 93.48%. The model achieved test accuracy of 95.70%. The confusion matrix of the results is illustrated in Figure 8. The results of this experiment are not worse than those of the previous experiment. The number of misclassifications is slightly higher as seen in the confusion matrix. The number of false negatives is smaller than in case of CNN without FFT. From the results, it seems that FFT preprocessing did not improve the final result.

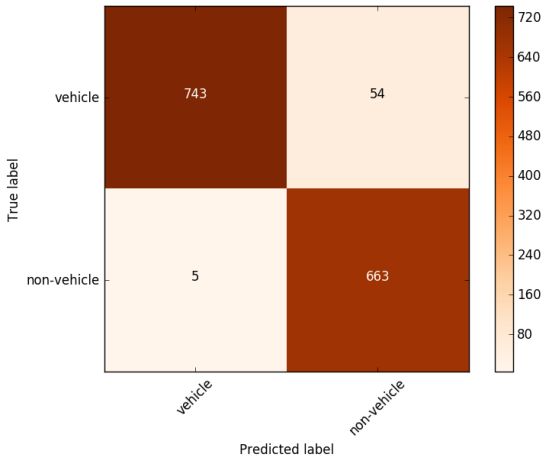

Fig. 7. Confusion matrix of the model without FFT preprocessing

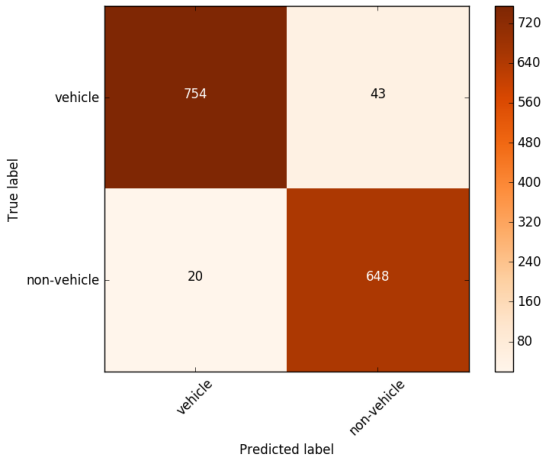

Fig. 8. Confusion matrix of the model with FFT preprocessing

V. Conclusion

Great results have been achieved in image classification and object detection by using convolutional neural networks. In this work, we applied such network to a rear-view vehicle images dataset. The preprocessing included data normalization and augmentation. In addition to that, another experiment was conducted by applying Fast Fourier Transform to images during preprocessing. From the results, we can conclude that the usage of Fourier Transform did not improve the final accuracy. On the contrary, using such preprocessing introduced an underfitting problem.

Future work could include a comparison of the proposed model's results with some existing convolutional neural networks. Additionally, investigating whether Fourier Transform could be used differently in the preprocessing step. Finally, find the solution for the underfitting problem.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.
[3] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[4] "The GTI-UPM Vehicle Image Database," 2018. [Online]. Available: http://www.gti.ssr.upm.es/data/
[5] J. Wang and L. Perez, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning."
[6] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, "Fourier transform," https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm.
[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556