Car model recognition with convolutional neural network

Barbara Koščević, Igor Aradski, Ema Smolić, Manuel Matošević, Filip Linarić and Maria Krajči

***Abstract*—In this paper we will be describing an implementation of car model recognition and classification using a convolutional neural network (CNN). CNNs can be used for numerous different tasks, such as image recognition, object localization, face recognition, in self-driving cars etc. Using a convolutional neural network ResNet-50, we implemented car model classification using the Stanford *Cars* dataset.**

***Index Terms*—CNN, vehicle detection, car model recognition, neural network**

I. Introduction

Convolutional neural network is the most modern neural network architecture primarily used for problems in the field of computer vision. CNN can be used for tasks such as image recognition, object localization, face recognition, in self-driving cars, etc. Using a convolutional neural network *ResNet-50* we built a model able to classify cars by their model. The network was trained on the Stanford Cars dataset. Object classification is a common task for neural networks. Car classification can help the police and owners of stolen cars locate the suspect's location by matching the model to the images of cars taken from major highway or road intersection. [6] As image recognition becomes faster and more accurate, it assumes even more important roles in maintaining public safety - e. g. by taking the perpetrator's license plate and car model and reporting them to the authorities in real time. [6] Besides public safety, image recognition can play an important role in marketing and sales. For example, car manufacturers can use image recognition technology on a car enthusiasts blog, Instagram or Twitter accounts and gauge consumer's sentiments towards a certain model and association between brands. Overall, image recognition has a wide range of applications that deserve to be further explored. [6]

II. Related work

In paper [6], the main focus is on comparing different neural network models for image recognition and determining the most accurate one. Authors use transfer learning to test the accuracy of five different models: VGG16, ResNet50, ResNet101, InceptionV3, Xception; all trained on Stanford Cars dataset. First they test the accuracy of basic CNN with five convolution layers with ReLU activation and Adam optimizer[1]. First results were a clear proof of overfitting. To reduce this, they added new layers and used regularization. The model was then having a small accuracy rate of 60% on training set and 15.6% on testing set. Testing accuracy of other models: VGG16 - 49.63%, RESNET-50 76.68%, RESNET-101 - 68.18%, INCEPTION V3 -73.46%, XCEPTION - 5.19%.

In [7] authors tested the accuracy of a neural network trained with ResNet150 on Stanford Cars dataset. Their model had test set accuracy over 80% and is used as a reference in the making of the model for our neural network.

III. Convolutional neural network

Neural networks are a subtopic of machine learning and the centre of deep learning algorithms. Networks consist of neuron layers which contain an input layer, at least one hidden layer and an output layer. The neurons are connected to other neurons on neighbouring layers and each connection has its weight. The neurons also possess a threshold value. If the output of any single neuron is above the specified threshold value, that neuron is activated and sends data to the next layer of the network. Otherwise, the data is not passed. There are different types of neural networks used for different tasks and data types. Convolutional neural networks are used for classification and computer vision tasks. Before CNN, manual, long-term feature extraction methods were used to identify objects in images. However, convolutional neural networks now provide a more scalable approach to image classification and object recognition problems using the principles of linear algebra, more precisely matrix multiplication, to recognize patterns within an image.

Convolutional neural networks are built of three main types of layers:

- convolutional layer
- pooling layer
- fully connected layer

The convolutional layer is the first layer of the convolutional network. While convolutional layers may be followed by additional convolutional or pooling layers, a fully connected layer is the final layer. With each layer, CNN grows in complexity and is able to identify larger parts of the image. Beginning layers are focused on simple features such as colors and edges. As the image data progresses through the layers of the CNN, it begins to recognize larger elements or shapes of the object until the model finally identifies the desired object. [6]

The convolutional layer is the basic building block of CNNs and most of the calculations are done by it. It requires several components, namely input data, filter and feature map. The

[1] https://keras.io/api/optimizers/adam/

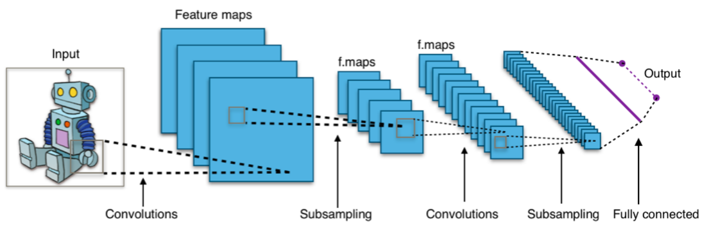


Fig. 1. Convolutional neural network architecture

Fig. 2. Example of car images from Stanford *Cars* dataset

convolution layer preserves the spatial relationship between pixels by learning image features using small squares of input data. Here the number of filters, filter size, step and zero padding are important parameters. After each convolution operation, CNN applies the ReLU activation function, introducing non-linearity into the model. Rectified linear unit (ReLU) is an element-based operation that replaces all negative pixel values in the feature map with zero. CNNs can use other nonlinear functions such as tanh or sigmoid, but ReLU is the default choice for development of multilayer perceptrons and convolutional neural networks. ReLU overcomes the problem of disappearing gradient, allowing models to learn faster and work better. [6]
The pooling layer is used to progressively reduce the size of the input display and reduce the feature dimension.
The name of the fully connected layer appropriately describes itself. In a fully connected layer, each neuron connects directly to all neurons in the next layer. This layer performs the classification task based on features extracted through the previous layers and their different filters. While convolution layers and pooling layers tend to use ReLU activation, fully connected layers typically use the softmax activation function, outputting a probability ranging from 0 to 1. [8] [7]

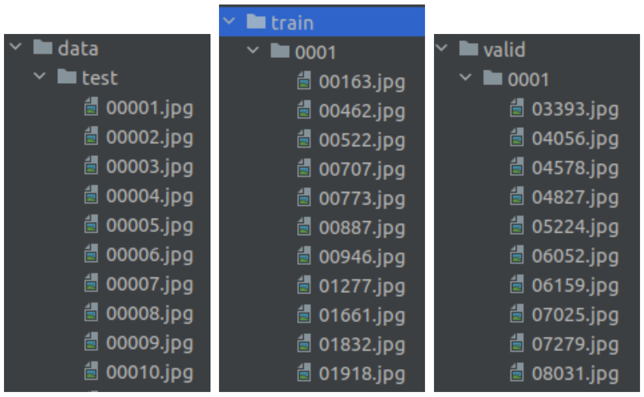
IV. Car model recognition

For the purpose of this project, the model was trained using the *ResNet-50* convolutional neural network over the *Cars* dataset.

A. Stanford Cars dataset

Cars dataset contains 16185 images of 196 different car models (classes). The data is divided into 8144 images in the training set and 8041 images in the test set, with each class divided approximately 50:50. Classes have features Make, Model and Year. [1] Our neural network model was trained and validated using this dataset.

While loading the dataset, three directories were created: *train*, *test* and *valid*. The *train* directory has images from the training set grouped by class. The *test* directory has images from the test set, while the *valid* directory contains a specific part of the training set images arranged in subdirectories representing classes to which they belong. File structure is shown on figure Fig.4.



```
data
  test
    00001.jpg
    00002.jpg
    00003.jpg
    00004.jpg
    00005.jpg
    00006.jpg
    00007.jpg
    00008.jpg
    00009.jpg
    00010.jpg
```

```
train
  0001
    00163.jpg
    00462.jpg
    00522.jpg
    00707.jpg
    00773.jpg
    00887.jpg
    00946.jpg
    01277.jpg
    01661.jpg
    01832.jpg
    01918.jpg
```

```
valid
  0001
    03393.jpg
    04056.jpg
    04578.jpg
    04827.jpg
    05224.jpg
    06052.jpg
    06159.jpg
    07025.jpg
    07279.jpg
    08031.jpg
```

Fig. 3. Directory organization *test, train and valid*

B. ResNet-50

ResNet, short for Residual Networks, is a classic neural network used as the basis for many computer vision tasks. Using it, one can train an extremely deep neural network with 150+ layers. Prior to ResNet, training very deep neural networks was difficult due to the gradient disappearance problem.
CNNs consist of different blocks, each block representing a group of operations which are applied to the input data. The identity block consists of a series of operations that retain the actual shape of the data, while the convolution block consists of a series of operations that reduce the shape or size of the data to a smaller one. CNN contains identity blocks and convolution blocks that reduce the input image to a compact set of numbers. Each of the obtained numbers should contain information useful to the classification of the image. Residual CNNs, such as ResNet-50, add an extra step to each block. The data is saved as a temporary variable before the operations that make the block are applied, and then this temporary data is added to the output data after the operations are finished. In general, this additional step is applied to each block.
To visualize this, imagine a set of n blocks represented by functions $f_1, f_2, \ldots, f_n$ and input data X. Classic CNN repetitively applies $f_n(f_{n-1} \ldots (f_3(f_2(f_1(X)))))$. Finally we get some Y_n which is the end result. The residual convolutional

neural network works on the following principle:

$$f_1(X) + X = Y_1$$
$$f_2(Y_1) + Y_1 = Y_2$$
$$....$$
$$f_n(Y_{n-1}) + Y_{n-1} = Y_n$$

where Y_n represents output of n-th block. [7]

C. Network Architecture

ResNet-50 is a CNN with 50 layers. The ResNet-50 model works in 5 steps, each with a convolution and identity block. Each convolution block and each identity block have 3 convolution layers respectively. The ResNet-50 has over 23 million trainable parameters. [3]

From the figure Fig.5. we can see that ResNet-50 architecture contains the following elements:

- A convolution with 64 different kernels with the kernel size of 7 * 7, all with a stride of size 2 giving us 1 layer.
- Max pooling with the stride size of 2.
- In the next convolution there is a 1 * 1,64 kernel followed with a 3 * 3,64 kernel and a 1 * 1,256 kernel at the end. These three layers are repeated in 3 times giving us 9 layers in this step.
- Furthermore, there is a kernel of 1 * 1,128 followed with a kernel of 3 * 3,128 and at last a kernel of 1 * 1,512. These three layers are repeated 4 time giving a total of 12 layers in this step.
- After that, there is a kernel of 1 * 1,256 and two more 3 * 3,256 kernels and 1 * 1,1024 which are repeated 6 time giving us a total of 18 layers.
- Again, there is a 1 * 1,512 kernel with two more of 3 * 3,512 and 1 * 1,2048. These are repeated 3 times giving us a total of 9 layers,
- After that, we do an average pool and end it with a fully connected layer containing 1000 nodes. Finally the softmax function adds one more layer.

We don't count the activation functions and the max/average pooling layers, which gives us a network with 1 + 9 + 12 + 18 + 9 + 1 = 50 layers.

D. Implementation

To implement our network we used *Keras*.[2] Keras is an open-source software library with *Python* inteface for artificial neural networks. It is used as an interface for *Tensorflow*[3] library. The program is divided into several parts. In general, there is a 'master file' in which parameter values are set and functions are called, as well as other files that contain the CNN architecture, methods, data reading methods, etc. In our project, the program is organised through the following files:

1) demonstrate.py - demonstrate work of neural network
2) mounting.py - file for loading data from dataset
3) resnet50.py - architecture of our network
4) train.py - training neural network
5) validate.py - model validation
6) test.py - testing neural network
7) identity_conv_block.py - convolution and identity block structure.

Input data (images from *Cars* dataset) is loaded with code in *mounting.py* and organised in directories as presented in paragraph *IV.B*. Prior to network training, preliminary training parameters were set. Variables *img_width* i *img_height* set dimensions of input data to 224x224. Variable *num_classes* is set to value 196, which is a number of classes our network has to classify input data . Variable *num_train_samples* set to 6515 indicates how much learning data will be used for training, while the variable *num_valid_samples* = 1629 indicates the amount of training data to be used for cross-validation. The variables *num_epochs* and *patience* determine how many times (number of epochs) the neural network will go through learning data, how many times we want the neural network to try to improve results, if there is no improvement in accuracy.
Another variable type used in this project are Callbacks[4], special utilities or functions that are performed during training in certain phases that help improve training performance or monitor the training process. They can help prevent overfitting, visualise learning progress, save checkpoints, generate logs, and more. Used callbacks are:

1) TensorBoard - visualization of training improvement
2) CSVLogger - print accuracy and cross-validation accuracy after every epoch
3) EarlyStopping - stops training if maximum accuracy is reached
4) ReduceLROnPlateau - used to increase learning rate
5) ModelCheckpoint - saves values of weights after every epoch

In *resnet50.py* transfer learning is applied. Transfer learning is a method used to improve neural network training. We added weights pretrained on *ImageNet*[5] dataset, which has 1000 classes, resulting in a fully connected layer which maps to 1000 nodes. The model obtained this way is used for training and testing. The optimizer used to create the network is *Standard gradient descent*[6] optimizer.
The model defined with the above parameters was trained through 100 epochs on the *Cars* dataset. Fig.5. shows the accuracy of the model in training and cross-validation, while Fig.6. shows the loss of the model.

Model was validated using data from validation set which includes 1629 images. Predictions were correct on 1422 images, giving the accuracy of 87.29%.

V. Results

Testing was performed using the model which was trained and 8041 images from the test set. 7016/8041 images were

[2]https://keras.io/
[3]https://www.tensorflow.org/

[4]https://www.tensorflow.org/api$_docs/python/tf/keras/callbacks/Callback$
[5]https://www.image-net.org/
[6]https://keras.io/api/optimizers/sgd/

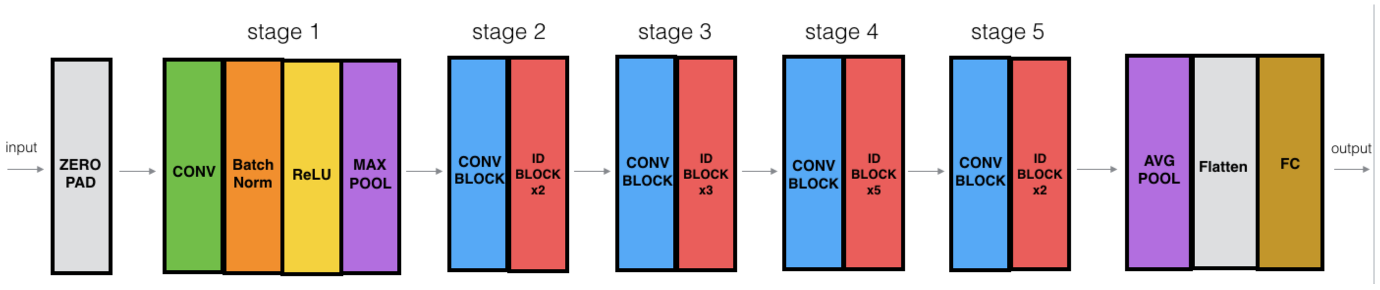


Fig. 4. ResNet-50 architecture overview

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 5. ResNet architecture depending on number of layers

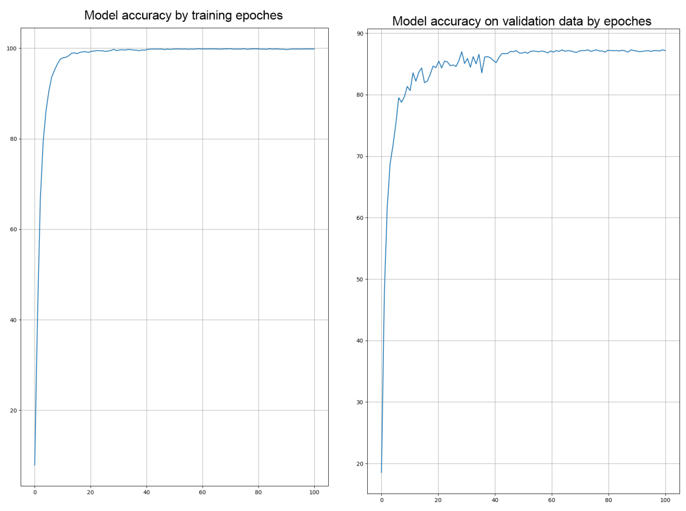


Fig. 6. Training and cross-validation accuracy

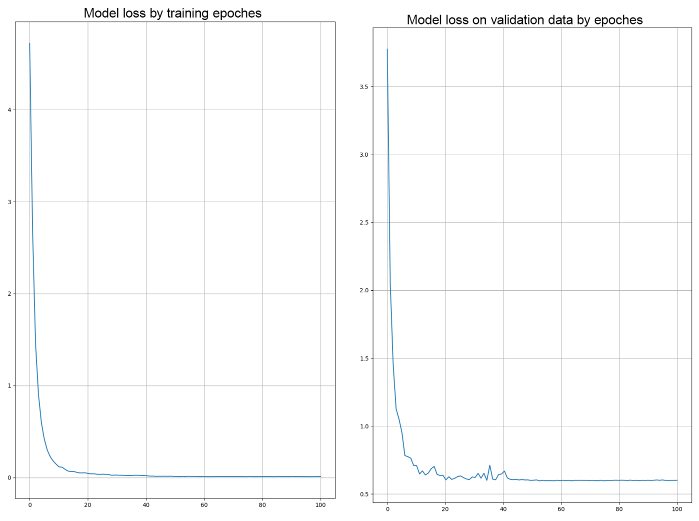


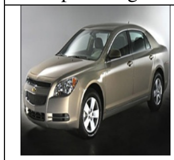
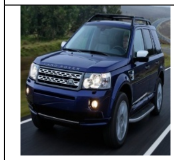
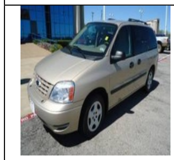
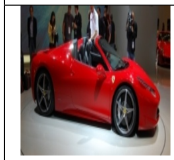
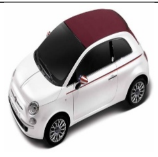
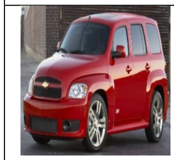
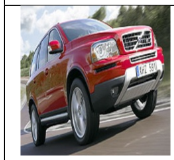
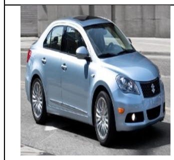
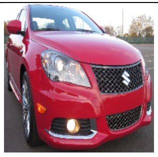
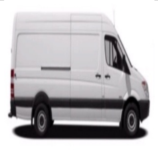
Fig. 7. Training and cross-validation loss

classified correctly, giving the accuracy of 87.25%. To demonstrate the performance, we used a set of 20 pictures from test set and used them as input. Examples of correct predictions are shown in Table I.

VI. Conclusion

Using CNNs over regular neural networks is advantageous for more accurate recognition of simple and complex patterns. Because of that, they are widely used in image recognition, image classification, image captioning and object detection. Recognising car models from images is a problem for which using a CNN is suitable. The results are satisfying but could be further improved by using a deeper CNN.

TABLE I
Results of demonstration

Input image	Prediction	Probability
	Chevrolet Malibu Hybrid Sedan 2010	0.9979
	Land Rover LR2 SUV 2012	0.9989
	Ford Freestar Minivan 2007	1.00
	Ferrari 458 Italia Convertible 2012	1.00
	FIAT 500 Convertible 2012	0.9507
	Chevrolet HHR SS 2010	1.00
	Volvo XC90 SUV 2007	0.9999
	Suzuki Kizashi Sedan 2012	0.9996
	Suzuki Kizashi Sedan 2012	1.00
	Dodge Sprinter Cargo Van 2009	0.5882

References

[1] J. Krause, M. Stark, J. Deng, and Li Fei-Fei, “3D Object Representations for Fine-Grained Categorization,” 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia, 2013

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, p. 770–778

[3] P. Dwivedi, “Understanding and Coding a ResNet in Keras,” https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

[4] B. Yerra, “Car Image Classification Using Features Extracted from Pre-trained Neural Networks,” https://towardsdatascience.com/classifying-car-images-using-features-extracted-from-pre-trained-neural-networks-39692e445a14

[5] Z. Dong, Y. Wu, M. Pei and Y. Jia, ”Vehicle Type Classification Using a Semisupervised Convolutional Neural Network,” in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 4, pp. 2247-2256, Aug. 2015, doi: 10.1109/TITS.2015.2402438.

[6] P. L. Schutter, “Car Image Recognition with Convolutional Neural Network Applications, ” https://patricia-schutter.medium.com/car-image-recognition-with-convolutional-neural-network-applications-e791c98c9d72

[7] E. Eames, H. Kropp, A Convolutional Neural Network Implementation For Car Classification “https://databricks.com/blog/2020/05/14/a-convolutional-neural-network-implementation-for-car-classification.html,”

[8] IBM Cloud Education, “Convolutional Neural Networks,” https://www.ibm.com/cloud/learn/convolutional-neural-networks

[9] Opengenus, ”Understanding ResNet50 architecture”, https://iq.opengenus.org/resnet50-architecture/