Faulty of Mathematics and Information Science
Warsaw University of Technology

Project documentation

Convolutional Neural Networks

Subject: Deep Learning

Authors:
Łukasz Pancer, Marcin Łukaszyk, Patryk Wrona

Warsaw
2022

Contents

1. Introduction

1.1. Description of the research problem

In this laboratory project we investigated different convolutional neural network architectures for image classification task. All of the experiments must have been conducted on CIFAR-10 dataset. [1] It contains 60 thousand of 32x32 color images in 10 classes. Each class has 6 000 images. Given are following classes: Airplane, Car, Bird, Cat, Deer, Dog, Frog, Horse, Ship and Truck. The dataset was split into 10 000 and 50 000 observations; in training and testing sets respectively.

In our work we checked the impact of different architectures of convolutional neural networks on the accuracy of predictions on test dataset. Besides, we compared different data augmentation techniques. Some pre-trained neural networks were taken into account to create the optimal model. Furthermore, we conducted research experiments for the optimal value of dropout parameter for an AlexNet pre-trained network, basic regularization techniques in standard convolutional neural network.

1.2. Instruction of the application (how to reproduce results)

In order to reproduce the results covered in this report, one must use python notebooks attached to this report:

— **Deep Learning Project 1.ipynb** - implementation and testing of different convolutional neural networks architectures, as well as training with the pre-trained networks
— **DL1_data-augmentation.ipynb** - notebook covering the comparison of different data augmentation techniques
— **DLAlexNetDrop_Out.ipnb** - notebook with a script for downloading data and model, and training with parameters defined in the experiment.

It is also important to use pre-trained neural networks from sources given in the first jupyter notebook, as well as the dataset Cifar-10 [1]. The experiments were conducted using Google Colab.

2. Theoretical background

2.1. Convolutional Neural Networks

Convolutional Neural network (CNN) is a type of artificial neural network most widely used with computer vision tasks. CNNs in contrast to regular deep neural networks preserve spatial information between features.

2.1.1. Convolutional layers

Convolutional layers are special type of layers designed for working with a two-dimensional image data. In this layers convolution is applied. Convolution is a special type of operation where for given part of an input a special mask is applied to calculate output value based on pixels affected by mask. It is widely used in computer vision to extract features from an image.

2.1.2. Pooling layers

Used to reduce the number of parameters. Mainly used as max-pool or average-pool where we assign one value from the squared part of input as a maximal value or an average of values.

2.1.3. Dense layers

The layer where each input is connected with each output similarly. For each output, we calculate the output value as a weighted sum of inputs, and bias, with an applied activation function.

2.2. Data Augmentation

Data Augmentation helps to overcome the huge demand for data of deep neural networks. It does so by extending an existing dataset with modified copies of contained samples, which not only makes data more extensive but can also compensate for uneven number of examples in each class.

2.2.1. Regular augmentation techniques

Rotation

Images can be rotated around a pivot.

Translation

Translation shifts an image horizontally or vertically. Similar to rotation empty space can be filled by reflection or a concealed part of image.

Shear

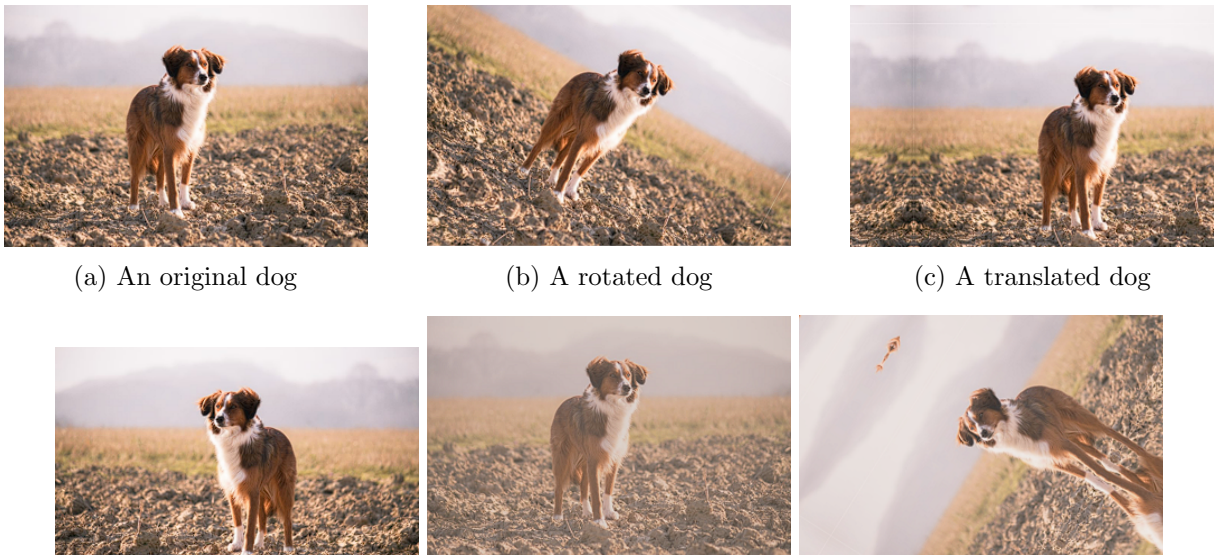
Shifting one part of an image in one direction, and the other one in the opposite one direction. It is similar to rotation, but the displacement is not around a pivot.

Flip

Flip could be performed vertically or horizontally. Flip creates a new image being a mirror of input image. Horizontal flip uses horizontal line in the very middle of the image as a mirror axis, while vertical flip uses vertical line.

Zoom

Transformation consisting of enlarging or narrowing an image while preserving its dimensions. In case of cooming in, the image becomes visually larger and in case of zooming out, the image becomes visually smaller and some pixels must be added (for example from the border of given image) to conserve the dimensions of given image.



(a) An original dog (b) A rotated dog (c) A translated dog

(d) A flipped dog (e) A dog with a modified contrast (f) A dog after all previous transformations

Figure 2.1: Basic transformations

2.2.2. Advanced augmentation techniques

Mixup

After chosing randomly a number from beta(here beta(0.5, 0.5)) distribution, 2 images are joined together by multiplying the pixels from the firs image by beta, while multiplying the pixels from the second image by (1 - beta). Then, new image is created and its label is also a result of weighted labels of given 2 images that mixup was used on.

2.3. Transfer Learning

Transfer learning is a technique that aims to take advantage of calculations already made on the problem similar to that in question. It makes use of high level features learned on a huge dataset, for example in tasks where amount of samples is greatly limited and not sufficient to correctly approximate the problem.

2.4. Ensemble Learning

Ensemble learning is a method of using multiple trained models to make predictions on data. The aim of creating ensemble learners is to make better predictions than the models from which given ensemble learner is composed. It is usual that creating ensemble learner is more time consuming than creating an individual model.

(a) A mix-up of dog (70%) and plane (30%)

Figure 2.2: Advanced transformation

3. Description of conducted experiments

3.1. Data augmentation

In order to test the impact of data augmentation techniques, given neural network architecture was used:

— **Input Layer** - 32x32x3
— **Dropout layer** - with rate 0.5
— **Flatten layer**
— **Dense layer 1** - 32 neurons
— **Dense layer 2** - 224 neurons
— **Dense layer 3** - 64 neurons
— **Dense layer 4** - 32 neurons
— **Dense layer 5** - 10 neurons, with softmax activation function

The network was trained using 20 000 iterations within 1 epoch for each data augmentation technique. Following data augmentation techniques were considered:

— **rotation** - up to 30 degrees clockwise and counter clockwise
— **translation** - up to 30% of image's width
— **shear** - up to 40 degrees clockwise and counter clockwise
— **flip** - horizontal and vertical flip, randomly, including both
— **zoom** - zoom in and out by up to 30%
— **mixup and rotation** - advanced data augmentation technique, additionally, rotations up to 10 degrees were performed on each augmented image

3.2. Dropout layer in AlexNet

In this experiment, we used pre-trained AlexNet and tested how well the network is performing based on paramiter of dropout layer. We used parameters of 0.0, 0.1, 0.3, 0.5, with 0.5 being default of AlexNet architecture.

We used pyTorch implementation using torch hub. We used a Stochastic Gradient Descent optimizer with a momentum of 0.9 and a learning rate of 0.001. For loss function we used cross-entropy. Each time we trained for 10 epochs. Each neural network was trained three times for every value of dropout. AlexNet is a convolutional neural network architecture designed in early 2010'. It achieved great results. It contains feature layers with 4 convolutional layers each with a max-pool layer and relu activation function.

3.3. ResNet50v2

For a network based on ResNet architecture we compared a vanilla implementation with Adam optimizer against the same model with a restarting cosine learning rate schedueler and weight clipping. The models were trained on augmented data.

3.4. Transfer Learning

Several pretrained architectures where tested. The two architectures covered the most were EfficientNetv2b0 and ResNet50v2. During Experiments it was found that these pretrained networks permorm poorly with images augmented by rotation. Random rotation of just 0.2 * 2PI in either side made both networks unable to converge and after 30 epochs of training the accuracy on the test set cycled seemingly random between 30 and 50 percent.

For both architectures the best accuracy was obtained with a Dense layer with activation, trained on data augmented by flipping, translating and changing contrast.

4. Results

4.1. Data augmentation techniques

The learning time of the model using data after horizontal and vertical flip was the lowest among all data augmentation techniques. The values of accuracy metric is low due to stopping the learning after 20 000 iterations within 1 epoch. The purpose of this experiment was to find the data augentation technique that leads to the best training (highest accuracy gain) after given (constant for all cases) amount of data. The results are following:

— **rotation** - Accuracy = 0.2184
— **translation** - Accuracy = 0.2090
— **shear** - Accuracy = 0.2137
— **flip** - Accuracy = **0.1977**
— **zoom** - Accuracy = 0.2151
— **mixup and rotation** - Accuracy = **0.2186**

The leading data augmentation technique is mixup with rotation - it received the highest value of accuracy after given number of iterations. The worst data augmentation technique was horizontal and vertical flip – the vlue of accuracy was particularly low (0.1977)

4.2. Dropout layer in AlexNet

Drop	Acc	Loss	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
0.5	84%	0.106	87%	91%	75%	78%	84%	75 %	81%	87%	88%	91%
0.0	84%	0.098	85%	90%	79%	69%	85%	77%	83 %	88%	91%	90%
0.1	82%	0.143	84%	89%	80%	66%	82%	78%	88%	87 %	86%	88%
0.3	81.3%	0.213	85%	90%	74%	75%	81%	72%	86 %	83%	83%	84%

For values of drop-out probability of 0 and 0.5 we achieved the best results with a similar Accuracy for whole testing dataset and for each individual class. For values of 0.1 and 0.3, we achieved worst values. The drop of accuracy is not consistent for every class with Dog, Ship and Truck losing a lot of accuracy with the worst value of drop-out probability and Bird, Plane and Car achieving similar results.

4.3. ResNet50v2

The vanilla network achieved an accuracy of 0.8373 after 24 epochs and at this points it stopped improving. After more than 100 epochs, network with restarting cosine scheduler and weight clipping got the best test set accuracy of 0.8505. The training accuracy was at about 93% at the time, and still viable to learn, but unfortunately the model along with training history were lost due to google colaboratory deciding to terminate the session because of the limits.

4.4. Transfer Learning

Not owning an Nvidia GPU, we were unable to conduct trainings multiple times with free Colaboratory limits. The validation on kaggle turned out to be difficult to conduct, as predicting 300000 images with the one of the following models on the CPU takes over 4 hours, and we have not found a way to upload the kaggle test set to Colaboratory any faster.

4.4.1. EfficientNetv2b0

EfficientNetv2b0 pretrained on ImageNet dataset got an accuracy of 0.8259 on the test set after 10 epochs. After additional 14 epochs with EfficientNet weights unfreezed and learning rate set to 0.00001, the accuracy got up to 84.21. Validation on Kaggle got the final score of 0.77170. The EfficientNet model was followed by a Dropout layer with coefficient of 0.5, a Dense layer with 256 nodes and a softmax layer. It was trained on non-augmented data.

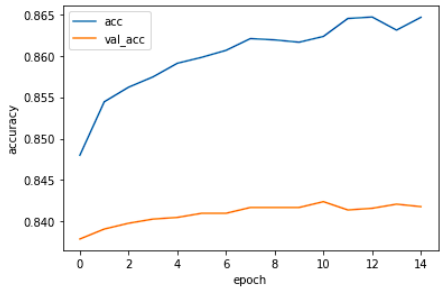


Figure 4.1: Initial 10 epochs of EfficientNet

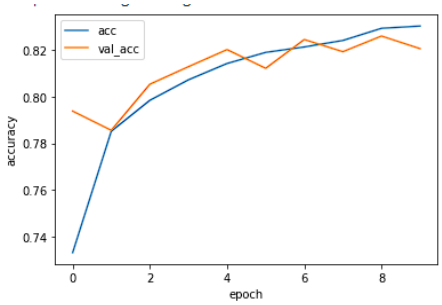


Figure 4.2: 14 epochs with weights unfreezed, EfficientNet

4.4.2. EfficientNetv2b0, augmented

The same network, but with ReLU activation after the dense layer, and trained on augmented data got an accuracy of 90.019 percent after 4 epochs.

4.4.3. ResNet50v2

ResNet50v2 with a Dense layer of 256 nodes and ReLU activation, Dropout of 0.5 and a softmax layer got 0.8963 accuracy after 15 epochs on augmented data. After additional 3 epochs with ResNet unfreezed accuracy on the test set increased to 0.9053. The model consisted of resizing layer, pretrained ResNet, dense layer with 256 nodes with ReLU activation and a softmax layer.

4.5. Ensemble

The Ensemble of the best performing EfficientNet and ResNet pretrained models got 0.9302 accuracy on test set without further training, being the best result that we achieved. Training for additional 3 epochs did not improve this result.

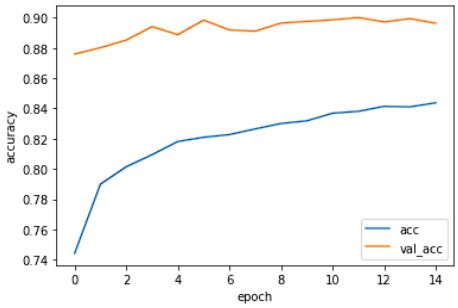


Figure 4.3: Initial 10 epochs of ResNet

5. Conclusions

5.1. Data augmentation techniques

From conducted experiments, one could deduce, that for this case, the best data augmentation techniques are mixup and rotation. They are the best methodologies to create new data of cifar-10 [1] to extend learning opportunities using convolutional learning networks. Image flips, altough being the easiest and the fastest data augmentation technique, it performed the worst, obtaining the lowest accuracy gain on test dataset.

5.2. Dropout layer in AlexNet

Dropout layers in AlexNet are essential for this architecture, as making sure that all of the parameters of segmentation layers are properly used in making a classification decision. Bigger values of drop-out probabilities can afflict how well a network can train in a given number of epochs, but it did not affect our experiment much, as results are presented with an increase in accuracy with a bigger parameter.

5.3. Transfer Learning

Transfer Learning proved to be a powerful utility, enabling us to get the best results. The pitfall is the amount of computation required to fine-tune the network. Models using transfer learning only took couple of epochs to get test accuracies of just below 90%, but a single batch of 32 images took up to a second to fit, depending on the model. The obtained results could still be improved upon as the training accuracy and loss for both models is still far below testing ones, but it would require additional hours of training.

5.4. Ensemble

The Ensemble of EfficientNet and ResNet performed surprisingly well. The testing accuracy is 93% obtained form an ensemble of two models with accuracy of about 90%. This proves really time efficient as training both models to 90% took only a fraction of time that rough extrapolating a training time to 93% of a single model yields

Bibliography

[1] Cifar-10 Dataset https://www.kaggle.com/c/cifar-10/
[2] ResNet50v2 https://tfhub.dev/google/imagenet/resnet_v1_50/feature_vector/5
[3] EfficientNetv2b0 https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_b0/feature_vector/2
[4] AlexNet https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

List of Figures

Dog - https://c1.peakpx.com/wallpaper/612/286/398/dog-pet-animal-hair-wallpaper-preview.jpg

List of Tables