DEEP LEARNING 2022

IMAGE CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS PROJECT NO. 1 PLAN

March 8, 2022

Elżbieta Jowik (298821), Agata Makarewicz (298827)

1 Dataset

A dataset used for this project, which is `CIFAR-10`, is an established computer-vision dataset for object recognition. The dataset contains of colour images with a (32×32) resolution. All of them are of shape $(32, 32, 3)$ where 3 represents the number of channels i.e R-G-B (Red, Green & Blue). The `10` in the set name refers to the cardinality of the set of unique values of the target variable

2 Data preparation

As mentioned before, the task underlying this project belongs to the class of multi-classification problems (`CIFAR-10` dataset has possible target 10 classes). Therefore, we find it inevitable to encode the target variable - the method we plan to employ is `One-Hot Encoding`.

In the raw data, all the image pixels are in a range from 0–255, therefore in order to accelerate the model training process, we consider the use of scaling all the pixel values to the range of $[0, 1]$ which is a standard procedure in images classification. Desired result can be obtained either by division of all the pixel values by 255 or the usage the built-in normalizing function.

For the purposes of the experiments, we also want to perform data augmentation, which is a technique used to alter existing images in order to create new training data. We plan to apply and compare common data augmentation methods such as cropping, scaling, rotating, stretching, and adding some noise to the picture.

3 Pre-trained Models and Existing Architectures

One of the objectives of this project is to utilize available pre-trained models. For this purpose, we have chosen `ResNet` and `VGG16` models pre-trained on `ImageNet` which is a dataset of over 14 million images belonging to 1000 classes.

4 Custom Architectures Details

Explanation of concepts in terms of functionality:

- `Conv2D`: a convolution layer used to extract features from the image or part of an image.
- `LeakyReLU`: applies Leaky version of a Rectified Linear Unit. It allows a small gradient when the unit is not active (allows to handle Dying ReLU problem).
- `Dropout`: applies Dropout to the input.
- `MaxPooling2D`: a pooling layer, that scales down the size of an image.
- `Flatten`: converts the n-dimensional array to a 1-dimensional array.
- `Dense`: a fully connected layer i.e. all the neurons in the current layer are connected to the next layer.
- `Activation`: a layer responsible for the application of an activation function.
- `BatchNormalization`: a layer that normalizes its inputs. Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

The idea behind all adopted approaches is to let the network learn of the simplest objects like lines or edges in the initial phase and then more complex ones which require more filters. All convolutional layers will use 3×3 kernels, but they are to differ as for the number of filters. Due to the fact that the `ReLu` activation function is not associated with the vanishing gradient problem, it will be applied in each layer of the network. To reduce the number of epochs relevant for learning the most accurate parameters we want include batch normalization in the experimental considerations. In turn to avoid or at least reduce the risk of an overfitting problem, we are will add dropouts.

Model concept no. 1	Model concept no. 2	Model concept no. 3
Layer type	Layer type	Layer type
Conv2D	Conv2D	Conv2D
LeakyReLU	Conv2D	MaxPooling2D
Conv2D	MaxPooling2D	Conv2D
LeakyReLU	Dropout	MaxPooling2D
MaxPooling2D	Conv2D	Flatten
Dropout	Conv2D	Dense
Conv2D	MaxPooling2D	Dropout
LeakyReLU	Dropout	Dense
Conv2D	Conv2D	Dropout
LeakyReLU	Conv2D	Dense
MaxPooling2D	MaxPooling2D	Dropout
Dropout	Dropout	
Flatten	BatchNormalization	
Dense	Flatten	
LeakyReLU	Dense	
Dropout	Dropout	
Dense	Dense	
Activation		

5 Learning process parametrization & evaluation

As an optimizer, we plan to use the ADAM method which is currently perceived as the most powerful one. We estimate that the learning process will require up to 200 epochs. However, to reduce the number of unnecessary steps (if needed) early stopping will be applied. As the most suitable loss function for the problem under consideration (for now) we find `categorical cross-entropy` loss function. Specific values of hyperparameters like `Batch size` or `learning rate` (among others) will be determined empirically through conducted experiments.

Models' performance will be evaluated on classification accuracy (the percent of labels that are predicted correctly) and the heatmap of the confusion matrix.