Machine Learning
Homework 2

Deadline: 7 March 2023

The exercise should be implemented in the Python programming language using Keras.

- Code snippets to aid you with your exercise can be found in e-class.
- Your final report should contain enough details to make your results reproducible.
- For exercise 3 you are free to use any publicly available data set as long as you specify what you have used. You may also augment any of these data sets as you see fit.

Installation

- The python installation procedure depends on your operating system. You can find information to help you with your installation online. Make sure you also install the pip tool. If you are using Ubuntu 18.04, python is already installed and pip can be installed with the comand "sudo apt-get install python-pip".
- Once you have installed python and pip, type in your terminal "pip install keras tensorflow matplotlib numpy". You can install any additional library using "pip install ⟨ library_name ⟩".
- Additional resources to help you get familiar with Python and Keras will be uploaded in e-class.

Exercise 1. *40%*

(A) Rewrite the general form of the backpropagation equations given in class for MLPs for the following specific activation functions i) ReLU, ii) hyperbolic tangent and iii) sigmoid. For each activation function write down the range of the gradients.

(B) Download the MNIST data set and use it to train a fully-connected neural network to recognise handwritten digits. The MNIST data set can be loaded from *keras.datasets*. Make sure that your data is normalized. Each hidden layer should have 32 units and the output layer should have a softmax activation function. Compile your model to use the standard SGD optimizer with a learning rate of 0.01 and the categorical crossentropy loss function. Use (i) 5, (ii) 20 and (iii) 40 layers. For each choice (i), (ii) and (iii), use the (a) ReLU, (b) hyperbolic tangent and (c) sigmoid activation function on all hidden layers. Report the test scores for each model. What are your observations?

(C) For each of the models in (B), trained for 3 epochs on the MNIST data set, compute for each layer the maximum value of the gradient on a given mini-batch and create a plot of "layer depth vs. max gradient". Organize your plots as a grid so that your results for the different activation functions for each depth choice appear on the same subplot. Can you explain your observations? What insight do you gain for the observations in (B)?

(D) Train a model using the topology given in (B) and the activation function

$$LeCun(x) = 1.7159\ tanh(\frac{2}{3}x) + 0.01x.$$

Compare the learning curves of the models using LeCun and hyperbolic tangent activation functions. Write down the backpropagation equations and the gradient range for the LeCun activation function. Plot the gradients for the choices of depth given above for an untrained model using LeCun and hyperbolic tangent activations.

Exercise 2. *20%*

[Adversarial Examples]

(A) Download the MNIST dataset (or simply load it from keras.datasets) and use keras to train, (ii) a convolutional neural network to classify hand-written digits. Remember to normalize your data.

Once you feel satisfied with your achieved results save your models to disk for later use. Call a MNIST-simple-generator $f(w) = x$, a function which given an input w outputs a 28x28 grayscale image. You can think of a MNIST-simple-generator as a neural network with one input neuron, always equal to 1, as a function of its parameters. In this exercise you will build MNIST-simple-generators to construct adversarial examples for your models from (A).

To understand what an adversarial example is, we restrain our attention to MNIST. For most digits in the dataset, your developed model is going to output the correct label with a high confidence. Given such a digit (that your model classifies correctly with high confidence) an adversarial example is constructed by applying small changes to the digit image (so small that they may be imperceptible to a human), which lead to our model miss-classifying the image. In fact, we can construct adversarial examples for any such digit and make our model predict for a given input whichever class we want. A lot of research has been done on building robust optimization methods that are not prone to adversarial example attacks.

(B) Choose your 5 favorite digits and take a sample for each from the MNIST dataset. Index these samples by i. Define a MNIST-simple-generator f_{ij} as a keras model, constructing adversarial examples around instance i for class j. Compile your model to use the Adam optimizer and the mean squared error loss. Define also the composite model $C \circ f$ (C is the model you built in (A)), making sure that before you compile you disable training for all parameters in C. (*Hint*: You may want to use the Keras functional API). For each sample you picked and for each class 0-10, train f_{ij} in two steps, repeated as many times as necessary. First train f_{ij} with target equal to sample i. Then, train the composite model with target equal the one-hot encoding of j. Plot your adversarial examples together with the original image on a grid.

(C) Repeat (A)-(B) on the CIFAR10 dataset, this time selecting only one image and compute adversarial examples for each class.

(D) (Bonus) Construct a new model which is robust to adversarial examples constructed in the above described way. Carefully explain your approach and validate on MNIST and CIFAR10.

Exercise 3. 40%

Train a classifier to do emotion recognition from face images. The emotions your model must detect are: 'anger', 'disgust', 'fear', 'happiness', 'sadness', 'surprise' and 'neutral'. You will find data at

https://www.kaggle.com/code/gauravsharma99/facial-emotion-recognition/data.

Your model should be able to process feed from you laptop camera in real time. Your model must be saved in .h5 format. You should provide a script that given an input camera device repeatedly collects frames and classifies them using your model printing the result on the screen.