Uppsala University

Introduction to Machine Learning, Big Data, and AI

Assignment 5

General information

- The recommended tool in this course is R (with the IDE R-Studio). You can download R **here** and R-Studio **here**.
- Report all results in a single, *.pdf -file. *Other formats, such as Word, Rmd, or similar, will automatically be failed.*
- The report should be submitted to the Studium.
- A report that do not contain the general information (see template) will be automatically rejected.
- When working with R, we recommend writing the reports using R markdown and the provided **R markdown template**. The template includes the formatting instructions and how to include code and figures.
- If you have a problem with creating a PDF file directly from R markdown, start by creating an HTML file, and then just print the HTML to a PDF.
- Instead of R markdown, you can use other software to make the pdf report, but the same instructions for formatting should be used. These instructions are also available in **the PDF produced from the R markdown template**.
- The course has its own R package `uuml` with data and functionality to simplify coding. To install the package just run the following:

  1. `install.packages("remotes")`
  2. `remotes::install_github("MansMeg/IntroML", subdir = "rpackage")`

- We collect common questions regarding installation, and technical problems in a course Frequently Asked Questions (FAQ). This can be found **here**.
- Deadline for all assignments is **Sunday at 23.59**. See the course page for dates.
- If you have any suggestions or improvements to the course material, please post in the course chat feedback channel, create an issue, or submit a pull request to the public repository!

1 Convolutional Neural Networks

1.1 A convolutional layer

As a first step we are going to implement a layer of a convolutional neural network with one filter. We will here not train the layer, just implement it to understand the inner workings.

Reading: Some good material for this exercise is Figure 9.1 in Goodfellow et al (2017) and the video Ng (2020).

Start by importing examples from the MNIST dataset as follows.

```
library(uuml)
data("mnist_example")
```

To visualize an image use:

```
im <- mnist_example[["5"]]
image(1:ncol(im), 1:nrow(im), im,
      xlab = "", ylab = "",
      xaxt='n', yaxt='n', main="4")
```

1. Visualize the the MNIST example digit 4.
2. Implement a convolution function called `convolution(X, K)` that takes an imput MNIST image (`X`), an arbitrary large square kernel (`K`) and returns a valid feature map. Below is an example of how it should work.

```
X <- mnist_example[["4"]][12:15,12:15]
X
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   56  250  116    0
## [2,]    0  240  144    0
## [3,]    0  198  150    0
## [4,]    0  143  241    0
```

```
K <- matrix(c(1, 1, 0, 0), nrow = 2)
K
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    1    0
```

```
convolution(X, K)
```

```
##      [,1] [,2] [,3]
## [1,]   56  490  260
## [2,]    0  438  294
## [3,]    0  341  391
```

3. Visualize the feature map of MNIST example digit 4 using the above 2 by 2 kernel `K`.

4. Now implement all steps in a `convolutional_layer(X, K, b, activation)` function that takes the kernel, bias and activation function. It should work as follows.

```
relu <- function(x) max(0, x)
X <- mnist_example[["4"]][12:15,12:15]
K <- matrix(c(1, 1, 1,
              0, 0, 0,
              0, 0, 0), nrow = 3, byrow = TRUE)
convolutional_layer(X, K, -370, relu)
```

```
##      [,1] [,2]
## [1,]   52    0
## [2,]   14   14
```

5. Run your convolutional layer on MNIST example digit 4 with bias -400. Visualize the feature map as you visualized the original image. What does the filter seem to capture?

6. Now transpose your filter and run your convolutional layer on MNIST example digit 4 with bias -450. Visualize the feature map. What does that transposed filter seem to capture?

7. As the last step in our convolutional layer, implement a 2 by 2, two stride max-pooling layer. It should work as follows.

```
X <- mnist_example[["4"]][12:15,12:15]
maxpool_layer(X)
```

```
##      [,1] [,2]
## [1,]  250  144
## [2,]  198  241
```

8. Now put it all together and visualize the final output of your own convolutional layer. Visualize the feature map.

```
X <- mnist_example[["4"]]
relu <- function(x) max(0, x)
K <- matrix(c(1, 1, 1,
              0, 0, 0,
              0, 0, 0), nrow = 3, byrow = TRUE)
output <- maxpool_layer(convolutional_layer(X, K, -370, relu))
```

1.2 Convolutional neural networks using Keras

We are now going to implement a convolutional neural network using Keras. Here Ch. 5.1 in Chollet and Allarie (2018) and the following tutorial might be useful. Remember

to load the `tesorflow` R package before loading ther `keras` R package. If you get stuck, ask for help from you fellow students in Slack.

1. Implement a Convolutional Neural Network for the MNIST dataset. The network should have two convolutional layers as follows.

```
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## conv2d (Conv2D)              (None, 26, 26, 32)        320
## _________________________________________________________________
## max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
## _________________________________________________________________
## conv2d (Conv2D)              (None, 11, 11, 32)        9248
## _________________________________________________________________
## flatten (Flatten)            (None, 3872)              0
## _________________________________________________________________
## dense (Dense)                (None, 64)                247872
## _________________________________________________________________
## dense (Dense)                (None, 10)                650
## =================================================================
## Total params: 258,090
## Trainable params: 258,090
## Non-trainable params: 0
```

2. Explain why there is 320 parameters in the first layer. How many are kernel weights (and why) and how many biases.

3. Train the network using Keras. What is your loss and accuracy on the MNIST dataset?

4. As the next step we are going to implement a similar network for the CIFAR-10 dataset using `dataset_cifar10()`. See the tutorial for details on how to load data. Implement a similar CNN as in the tutorial, that is:

```
## Model: "sequential"
## _________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =================================================================
## conv2d (Conv2D)              (None, 30, 30, 32)        896
## _________________________________________________________________
## max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
## _________________________________________________________________
## conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
## _________________________________________________________________
## max_pooling2d_1 (MaxPooling2D) (None, 6, 6, 64)        0
## _________________________________________________________________
## conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
## _________________________________________________________________
## flatten (Flatten)            (None, 1024)              0
## _________________________________________________________________
## dense (Dense)                (None, 64)                65600
## _________________________________________________________________
```

```
## dense_1 (Dense)                 (None, 10)                650
## ================================================================================
## Total params: 122,570
## Trainable params: 122,570
## Non-trainable params: 0
## --------------------------------------------------------------------------------
```

5. Why do we now have 896 parameters in the first convolutional layer?

6. Try out at least 3 different networks, describe the networks, why you chose it and the keras model output. For example, you can try to add a `dropout_layer` (see Chollet and Allaire, 2018, Ch. 5.3). Can you improve over the previous network with respect to accuracy? You can also use techniques from the previous lab.

1.3 Transfer learning using VGG16

As a last step we will look into using transfer learning as a quick way of improving the prediction accuracy on the cifar-10 dataset. Here Ch. 5.3 in Chollet and Allaire (2018) or this tutorial might be helpful.

1. Using keras, download the VGG16 convolutional neural network. Just download the covolutional base. We are going to use the network on the cifar-10 dataset so change the `input_size` to `c(32, 32, 3)`.

2. Now add a dense layer with 64 hidden nodes (as in the previous exercise). Include the keras model output in the assignment report. How many parameters do you have in the top dense layer (compared to the CNN in the previous part).

3. Now, freeze the convolutional base and train the top dense layer of your network on the cifar-10 dataset. Run it for 5 epochs. *Note!* This will take time. Don't be surprised if you need roughly 200s per epoch.

4. Report your final accuracy. How does this compare to the previous model for the cifar data?