Lab 7: Neural Networks

Dataset

In this lab, we'll build a model for predicting if we have an image of a dog or a cat. For this, we will use the "Dogs & Cats" dataset that can be downloaded from [Kaggle](https://www.kaggle.com/c/dogs-vs-cats/data).

You need to download the `train.zip` file.

If you have trouble downloading from Kaggle, use [this link](https://github.com/fenago/large-datasets/releases/download/dogs-cats/train.zip) instead:

```bash
wget https://github.com/fenago/large-datasets/releases/download/dogs-cats/train.zip
```

In the lectures we saw how to use a pre-trained neural network. In the lab, we'll train a much smaller model from scratch.

Note: You don't need a computer with a GPU for this lab. A laptop or any personal computer should be sufficient.

Data Preparation

The dataset contains 12,500 images of cats and 12,500 images of dogs.

Now we need to split this data into train and validation

* Create a `train` and `validation` folders
* In each folder, create `cats` and `dogs` folders
* Move the first 10,000 images to the train folder (from 0 to 9999) for boths cats and dogs - and put them in respective folders
* Move the remaining 2,500 images to the validation folder (from 10000 to 12499)

You can do this manually or with Python (check `os` and `shutil` packages).

Model

For this lab we will use Convolutional Neural Network (CNN. Like in the lectures, we'll use Keras.

You need to develop the model with following structure:

* The shape for input should be `(150, 150, 3)`
* Next, create a covolutional layer ([`Conv2D`](https://keras.io/api/layers/convolution_layers/convolution2d/)):
    * Use 32 filters
    * Kernel size should be `(3, 3)` (that's the size of the filter)
    * Use `'relu'` as activation
* Reduce the size of the feature map with max pooling ([`MaxPooling2D`](https://keras.io/api/layers/pooling_layers/max_pooling2d/))
    * Set the pooling size to `(2, 2)`
* Turn the multi-dimensional result into vectors using a [`Flatten`](https://keras.io/api/layers/reshaping_layers/flatten/) layer
* Next, add a `Dense` layer with 64 neurons and `'relu'` activation
* Finally, create the `Dense` layer with 1 neuron - this will be the output
    * The output layer should have an activation - use the appropriate activation for the binary classification case

As optimizer use [`SGD`](https://keras.io/api/optimizers/sgd/) with the following parameters:

* `SGD(lr=0.002, momentum=0.8)`

For clarification about kernel size and max pooling, check [Week #11 Office Hours](https://www.youtube.com/watch?v=1WRgdBTUaAc).

Question 1

Since we have a binary classification problem, what is the best loss function for us?

Note: since we specify an activation for the output layer, we don't need to set `from_logits=True`

Question 2

What's the total number of parameters of the model? You can use the `summary` method for that.

Generators and Training

For the next two questions, use the following data generator for both train and validation:

```python
ImageDataGenerator(rescale=1./255)
```

* We don't need to do any additional pre-processing for the images.
* When reading the data from train/val directories, check the `class_mode` parameter. Which value should it be for a binary classification problem?
* Use `batch_size=20`
* Use `shuffle=True` for both training and validaition

For training use `.fit()` with the following params:

```python
model.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=50
)
```

Note `validation_steps=50` - this parameter says "run only 50 steps on the validation data for evaluating the results".
This way we iterate a bit faster, but don't use the entire validation dataset.
That's why it's important to shuffle the validation dataset as well.

Question 3

What is the median of training accuracy for this model?

Question 4

What is the standard deviation of training loss for this model?

Data Augmentation

For the next two questions, we'll generate more data using data augmentations.

Add the following augmentations to your training data generator:

* `rotation_range=40,`
* `width_shift_range=0.2,`
* `height_shift_range=0.2,`
* `shear_range=0.2,`
* `zoom_range=0.2,`
* `horizontal_flip=True,`
* `fill_mode='nearest'`

Question 5

Let's train our model for 10 more epochs using the same code as previously.
Make sure you don't re-create the model - we want to continue training the model we already started training.

What is the mean of validation loss for the model trained with augmentations?

Question 6

What's the average of validation accuracy for the last 5 epochs (from 6 to 10) for the model trained with augmentations?