Chapter 14: Programming Exercise CIFAR10

Exercise: Classification with a CNN

In this exercise we are going to classify images of airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships and trucks.

Step 1: Load the Data

- We start by reading the dataset we are going to use. This is a built-in dataset in tensorflow_datasets. You get this code in the cells below.

```
from keras.datasets import cifar10
(X_train_full, y_train_full), (X_test, y_test) = cifar10.load_data()
```

- How many examples does the training set and test set contain?
- We will use the last 10000 examples from the training set as our validation set. The other examples will be used as training set. Write code to create the validation and training data.

Step 2: Data exploration

- Give the number of samples for each category.
- Show the first 25 examples in a 5 by 5 grid, each time showing the class (label) of each example along with the image.

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'shi
```

Step 3: Build a Model

- Build a model using the Sequential API. Use the following architecture:
  - A `Rescaling` layer which rescales the inputs (that are now between 0 and 255) to the range $[0, 1]$.
  - A `Convolutional` layer with 32 filters, kernel size equal to (3, 3), padding=`same` and the `relu` activation function
  - A `Convolutional` layer with 32 filters, kernel size equal to (3, 3), padding=`same` and the `relu` activation function
  - A max pooling layer with pool size equal to (2, 2).
  - A `Convolutional` layer with 64 filters, kernel size equal to (3, 3), padding=`same` and the `relu` activation function
  - A `Convolutional` layer with 64 filters, kernel size equal to (3, 3), padding=`same` and the `relu` activation function
  - A max pooling layer with pool size equal to (2, 2).
  - A `Flatten` layer that transforms the tensors to 1 long 1D tensor
  - A `Dense` layer with 128 units and the `relu` activation function.
  - A `Dense` output layer.
    - How many units should this layer have?

      * What is the most appropriate activation function for this layer, given that we are doing classification into 10 classes?

- Write a function `get_model()` that returns this model.
- Ask for the `summary` of the returned model.

Step 4: Compile the Model

Next, compile the model.

- Use `adam` as the optimizer with learning_rate = 0.001.
- Specify the correct loss for this classification problem.
- Track the `accuracy` metric.

Step 5: Train the Model

- Train the model.
  - Train it for a large number epochs.
    * Consider using the early stopping callback.
  - Use a batch size of 32.
  - Be sure to use the validation data.
- Use the `history` object returned by the `fit` method to plot the learning curves. You can use the code given below to plot the learning curves.

```python
def plot_learning_curves(history):
    plt.figure(figsize=(8, 5))
    for key, style in zip(history.history, ["r-o", "r-*", "b-o", "b-*"]):
        epochs = np.array(history.epoch)
        plt.plot(epochs + 1, history.history[key], style, label=key)
    plt.xlabel("Epoch")
    plt.axis([1, len(history.history['loss']), 0., 1])
    plt.legend(loc="lower left")
    plt.grid()
```

Step 6: Evaluate the Model

- Use the `evaluate` method to evaluate the model on the test set.
  - What is the performance of the model on the test set?
- We now look at the individual predictions on the test set, in particular we will look for misclassified images.
  - Create an array `predictions` with the prediction according to the model. Note: the `predict` method returns 3 probabilities for each image, i.e. one probability per class.
  - Now go over the test set and compare the predicted labels with the expected labels. When there is a difference between predicted and expected label, you add the image to a list `incorrect_predicted_images` and you collect the correct and

(incorrectly) predicted labels in two lists `expected_labels` and `predicted_labels`.

- Create the confusion matrix
- Show the first 25 examples of the test set that were wrongly predicted. Use a 5 by 5 grid, each time showing the predicted label along with the expected label.

Step 7: Improve the Model

- Try to improve the model:
  - Does adding an additional "convolutional block" improve the model?
  - Does changing the `kernel_initializer` improve the model? Since we are using `relu` activation functions, it is recommended (see chapter 11) to use `he` initialization.
  - Does using dropout improve the model? Dropout is a regularization technique that can be used to improve the performance of the model. See chapter 11 for more information, more in particular section 11.5.2
  - Does using a different optimizer or learning rate improve the model?
  - ....