Tutorial

Send an email at the following address: [samuel.rochette@epfedu.fr](http://samuel.rochette@epfedu.fr)

It must contain a HTML export of the python notebook you’ll create.

Open a new python notebook for each of the 6 sessions we will have together. Please use markdown and make it comfortable to read.

# Visualize the activations

In this part, we will see how the neural net sees the image in each layer. You’ll need your trained model from the previous tutorial.

Take your tutorial 3 notebook and do as follow:

* classifier.save("model.h5")

Then in your tutorial 4 notebook, load it:

* model = tf.keras.models.load\_model(“model.h5”)

1. Load an image of your choice. Resize it to match the input height and width of your model. Add one dimension for the batch. Shape must then be (1, height, width, 3)
2. Define the following function

| def plot\_activation(activation, plot\_height, plot\_width):  ix = 1  for \_ in range(plot\_height):  for \_ in range(plot\_width):  # specify subplot and turn of axis  ax = plt.subplot(plot\_height, plot\_width, ix)  ax.set\_xticks([])  ax.set\_yticks([])  # plot filter channel in grayscale  plt.imshow(activation[0, :, :, ix - 1], cmap='gray')  ix += 1  # show the figure  plt.show() |
| --- |

1. Try to understand the function behaviour and give it the following arguments (change the number 12 in “layer\_outputs” accordingly if you modified the architecture):

| from tensorflow.keras import models # Extracts the outputs of the top 12 layers layer\_outputs = [layer.output for layer in model.layers[:12]] # Creates a model that will return these outputs, given the model input activation\_model = models.Model(inputs=model.input, outputs=layer\_outputs) # Returns a list of five Numpy arrays: one array per layer activation activations = activation\_model.predict(my\_image) # plot the first activation layer (4x8 => 32 filters or activations) plot\_activation(activations[0], 4, 8) |
| --- |

1. Plot all the other activation layers (adapt “plot\_height” and “plot\_width” parameters)
2. For the last layers (from Flatten to the end), you must use another function that plots scalars instead of matrix:

| def plot\_neuron(neuron, number):  ix = 1  vmin = min(neuron[0, :number])  vmax = max(neuron[0, :number])  for \_ in range(number):  # specify subplot and turn of axis  ax = plt.subplot(1, number, ix)  ax.set\_xticks([])  ax.set\_yticks([])  # plot filter channel in grayscale  plt.imshow([neuron[:, ix - 1]], cmap="gray", vmin=vmin, vmax=vmax)  ix += 1  plt.show() |
| --- |

1. Interpret and define the plot. What is learnt by the model? How does it evolve when you go deeper?

# Serve your model

For this part, please attach to the mail the python file where the app is coded. Also, add a screenshot of your running app in the html notebook.

1. Save your trained model to “h5” format
2. Create a small streamlit webapp
3. Add an option to upload an image (you can use: decoded = cv2.imdecode(np.frombuffer(image\_bytes, np.uint8), -1))
4. Preprocess the image, run the inference and display the classification result
5. Take a photo with your computer camera, preprocess the image and run the inference on it
6. Imagine a process to get feedback on the model prediction and enhance the performance for the next user. Implement it.