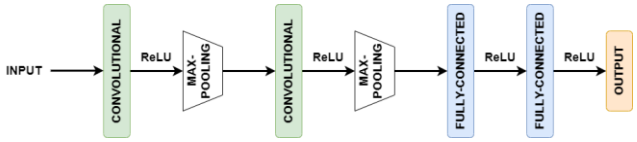
Assignment 2: Convolutional neural networks *(approximate training run-time on a CPU: 40 min)*

Implement a convolutional neural network that solves the speech classification task. This neural network must have two convolutional layers with max-pooling followed by two fully-connected layers with 128 neurons each. Apply a rectified linear unit (ReLU) activation function at the output of each convolutional and fully-connected layer. Furthermore, the first and second convolutional layers use 32 and 16 feature maps, respectively. The kernel size of both of them is 5x5, the stride is 1x1 and no padding is considered. Pooling size is 2x2 and the stride for max-pooling is 2x2.



Again, consider a minibatch size of 1,024 samples and, as the optimizer, employ stochastic gradient descent with a learning rate of 0.01, momentum of 0.9 and weight decay of 0.00001. Train your model for 50 epochs. Use "torch.manual_seed(0)" to make your results reproducible.

1) What is the size of your input layer and why?
2) As you know, prior to feed the output of the second max-pooling layer to the first fully-connected layer, you have to "flatten" (that is, reshape) that output, which is a volume, so all of its elements are re-arranged into a vector. In your code, calculate the length of this vector as a function of the input size, kernel size, pooling size and number of feature maps. Store the result in a variable called "flattened_size" and use it to define the first fully-connected layer. What is the value of "flattened_size"? Report the snip of code that you wrote to calculate "flattened_size".
3) Plot the training and validation losses, as well as the training and validation accuracies, as a function of the training iteration. Once done, re-train your model from scratch by using Adam with default parameters (instead of stochastic gradient descent) as the optimizer and plot the same types of curves. Compare the results very briefly.
4) What is the test accuracy from using stochastic gradient descent? And from using Adam?
5) Calculate, by hand, the total number of parameters of the model, and indicate, step by step, how you reached your solution.
6) Very briefly, compare, in terms of performance and computational complexity, this model with the feedforward neural network that you implemented in the previous assignment.

Hints:

- Do not forget to normalize your speech feature matrices, e.g., `X_train ← (X_train - np.mean(X_train)) / np.std(X_train)`
- Shuffle your training data before model training.
- Accuracy is defined as the ratio between the number of correct classifications and the total number of classifications.
- Training on a CPU takes time. To check more quickly that your implementation is correct, you may want to train your model for 5 epochs using Adam, as well as considering 8 and 4 feature maps for the first and second convolutional layers, respectively. In this case, test accuracy should be *around* 65%.
- To check that you correctly calculated by hand the number of parameters of your model, you may want to use the following snip of code:

```
def get_no_params(model):
    nop = 0
    for param in list(model.parameters()):
        nn = 1
```

```
    for s in list(param.size()):
        nn = nn * s
    nop += nn
return nop
```