**Overfitting detection**

In this exercise, we'll work with a small subset of the examples from the original sign language letters dataset. A small sample, coupled with a heavily-parameterized model, will generally lead to overfitting. This means that your model will simply memorize the class of each example, rather than identifying features that generalize to many examples.

You will detect overfitting by checking whether the validation sample loss is substantially higher than the training sample loss and whether it increases with further training. With a small sample and a high learning rate, the model will struggle to converge on an optimum. You will set a low learning rate for the optimizer, which will make it easier to identify overfitting.

Note that keras has been imported from tensorflow.

**Instructions**

**100 XP**

* Define a sequential model in keras named model.
* Add a first dense layer with 1024 nodes, a relu activation, and an input shape of (784,).
* Set the learning rate to 0.01.
* Set the fit() operation to iterate over the full sample 200 times and use 50% of the sample for validation purposes.

# Define sequential model

model = keras.Sequential()

# Define the first layer

model.add(keras.layers.Dense(1024, activation='relu', input\_shape=(784,)))

# Add activation function to classifier

model.add(keras.layers.Dense(4, activation='softmax'))

# Finish the model compilation

model.compile(optimizer=keras.optimizers.Adam(lr=0.01),

loss='categorical\_crossentropy', metrics=['accuracy'])

# Complete the model fit operation

model.fit(sign\_language\_features, sign\_language\_labels, epochs=200, validation\_split=0.50)

Excellent work! You may have noticed that the validation loss, val\_loss, was substantially higher than the training loss, loss. Furthermore, if val\_loss started to increase before the training process was terminated, then we may have overfitted. When this happens, you will want to try decreasing the number of epochs.