**Metrics and validation with Keras**

We trained a model to predict sign language letters in the previous exercise, but it is unclear how successful we were in doing so. In this exercise, we will try to improve upon the interpretability of our results. Since we did not use a validation split, we only observed performance improvements within the training set; however, it is unclear how much of that was due to overfitting. Furthermore, since we did not supply a metric, we only saw decreases in the loss function, which do not have any clear interpretation.

Note that keras has been imported for you from tensorflow.

**Instructions**

**100 XP**

* Set the first dense layer to have 32 nodes, use a sigmoid activation function, and have an input shape of (784,).
* Use the root mean square propagation optimizer, a categorical crossentropy loss, and the accuracy metric.
* Set the number of epochs to 10 and use 10% of the dataset for validation.

# Define sequential model

model = keras.Sequential()

# Define the first layer

model.add(keras.layers.Dense(32, activation='sigmoid', input\_shape=(784,)))

# Add activation function to classifier

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

# Set the optimizer, loss function, and metrics

model.compile(optimizer='RMSprop', loss='categorical\_crossentropy', metrics=['accuracy'])

# Add the number of epochs and the validation split

model.fit(sign\_language\_features, sign\_language\_labels, epochs=10, validation\_split=0.10)

Nice work! With the keras API, you only needed 14 lines of code to define, compile, train, and validate a model. You may have noticed that your model performed quite well. In just 10 epochs, we achieved a classification accuracy of around 98% in the validation sample!