**Course: Artificial Neural Networks and Deep Learning**

**Exercises Unit 2: Deep Learning Methods**

**Exercise 1. Performance evaluation**

Determine the following values: train error, test error (using dev set), and training time. Repeat the execution at least three times. **Exercise 2: Changing basic hyperparameters**

Change hyperparameters related to: batch size, number of layers, and number of neurons. Estimate train error, test error and training time.

**Exercise 3: Changing activation functions**

Change the activation function in the following part of the code:

| for neurons in n\_neurons\_per\_hlayer:  model.add(keras.layers.Dense(neurons, activation="relu") |
| --- |

Possible activation functions: elu, relu, tanh, linear

(Other functions: sigmoid, softsign, softplus, hard\_sigmoid, exponential, softmax, selu)

More information about activation functions: https://conx.readthedocs.io/en/latest/ActivationFunctions.html **Exercise 4: Changing initializers**

Change the lines:

| my\_initializer= keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None) for neurons in n\_neurons\_per\_hlayer:  model.add(keras.layers.Dense(neurons, activation="relu", kernel\_initializer=my\_initializer)) |
| --- |

Possible initializers:

| keras.initializers.Zeros()  keras.initializers.RandomUniform(minval=-0.05, maxval=0.05, seed=None)  keras.initializers.RandomNormal(mean=0.0, stddev=0.05, seed=None)  keras.initializers.he\_uniform(seed=None)  keras.initializers.he\_normal(seed=None) |
| --- |

See this link for more information about initializers: https://keras.io/initializers/

**Exercise 5: Using batch normalization**

Change the lines where the model is created:

Normalize after activation function:

| model = keras.models.Sequential([  keras.layers.InputLayer(input\_shape=(INPUTS,), batch\_size=None),  keras.layers.Dense(500, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.BatchNormalization(),  keras.layers.Dense(250, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.BatchNormalization(),  keras.layers.Dense(75, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.BatchNormalization(),  keras.layers.Dense(25, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.BatchNormalization(),  keras.layers.Dense(3, activation="softmax")]) |
| --- |

Normalize before activation function:

| model = keras.models.Sequential([  keras.layers.InputLayer(input\_shape=(INPUTS,), batch\_size=None),  keras.layers.Dense(500, kernel\_initializer="he\_normal", use\_bias=False),  keras.layers.BatchNormalization(),  keras.layers.Activation("elu"),  keras.layers.Dense(250, kernel\_initializer="he\_normal", use\_bias=False),  keras.layers.BatchNormalization(),  keras.layers.Activation("elu"),  keras.layers.Dense(75, kernel\_initializer="he\_normal", use\_bias=False),  keras.layers.BatchNormalization(),  keras.layers.Activation("elu"),  keras.layers.Dense(25, kernel\_initializer="he\_normal", use\_bias=False),  keras.layers.BatchNormalization(),  keras.layers.Activation("elu"),  keras.layers.Dense(3, activation="softmax")]) |
| --- |

**Exercise 6: Using regularization**

In this exercise, analyze two different cases: (a) batch size=512 and (b) batch size=256.

PART (1): Use regularizer L2 as follows:

| model = keras.Sequential(name="my\_model")  model.add(keras.layers.InputLayer(input\_shape=(INPUTS,), batch\_size=None))  for neurons in n\_neurons\_per\_hlayer:  model.add(keras.layers.Dense(neurons, activation="relu",  kernel\_regularizer=keras.regularizers.l2(0.001)))  model.add(keras.layers.Dense(OUTPUTS, activation="softmax",  kernel\_regularizer=keras.regularizers.l2(0.001))) |
| --- |

Possible regularizers:

| keras.regularizers.l1(0.001)  keras.regularizers.l2(0.001) |
| --- |

Analyze regularizers with different values for lambda (e.g., 0.001, 0.0001, …)

PART (2): Use dropout as follows:

| model = keras.models.Sequential([  keras.layers.InputLayer(input\_shape=(INPUTS,), batch\_size=None),  keras.layers.Dense(500, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.Dropout(rate=0.01),  keras.layers.Dense(250, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.Dropout(rate=0.01),  keras.layers.Dense(75, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.Dropout(rate=0.01),  keras.layers.Dense(25, activation="elu", kernel\_initializer="he\_normal"),  keras.layers.Dropout(rate=0.01),  keras.layers.Dense(3, activation="softmax")]) |
| --- |

Analyze different values of dropout rate (e.g., rate = 0.1, 0.01, …)

**Exercise 7: Changing learning rate and epochs**

Change learning rate and number of epochs. Their values are written in the following lines of code:

| n\_epochs = 1000 |
| --- |

| learning\_rate = 0.1 |
| --- |

**Exercise 8: Changing optimizers**

Change optimizer:

| model.compile(loss=tf.keras.losses.categorical\_crossentropy,  optimizer=tf.keras.optimizers.SGD(lr=learning\_rate),  metrics=["categorical\_accuracy"]) |
| --- |

Optimizers:

| tf.keras.optimizers.SGD(lr=learning\_rate, momentum=0.9)  tf.keras.optimizers.SGD(lr=learning\_rate, momentum=0.9,nesterov=True)  tf.keras.optimizers.RMSprop(lr=learning\_rate, rho=0.9)  tf.keras.optimizers.Adam(lr=learning\_rate, beta\_1=0.9, beta\_2=0.999) |
| --- |

NOTE: Try also smaller learning rates for RSMprop and Adam (0.01, 0.001, etc.)

Other optimizers: https://www.tensorflow.org/api\_docs/python/tf/keras/optimizers

**Exercise 9: Multiple changes**

Combine multiple changes of hyperparameters to obtain the best performance.