Model architecture:

#%%writefile ""/content/drive/My Drive/ML/ML\_experiments/Trained\_Models/CNN\_ModelsModel\_\_2020\_19\_01\_\_09\_41\_59\_\_410624/model\_settings.text

#%%writefile $path/model\_settings.text

# Line above: Save model settings to file for reproducability - Run once with command above and once without.

# First, it saves cell's content, but doesn't run the cell, afterwards, it's running the cell

# Reset tf sessions

tf.keras.backend.clear\_session() # Destroys the current TF graph and creates a new one.

dimensions = 512 # Image-Dimension: 512x512x1 (BW-image)

classes = 10

# Set up model architecture in terms of its layers

model = models.Sequential()

# Set layers

model.add(layers.Conv2D(32, (3, 3), activation='relu', strides=1, input\_shape=(dimensions, dimensions, 1), # 32 batch size

kernel\_regularizer=regularizers.l2(0.1)

))

model.add(layers.Conv2D(32, (3, 3), activation='relu', strides=2, kernel\_regularizer=regularizers.l2(0.1)

))

model.add(layers.Conv2D(32, (3, 3), activation='relu', strides=2, kernel\_regularizer=regularizers.l2(0.1)

))

model.add(layers.Conv2D(4, (5, 5), activation='relu', strides=3, kernel\_regularizer=regularizers.l2(0.1)

))

model.add(layers.Conv2D(4, (3, 3), activation='relu', strides=2, kernel\_regularizer=regularizers.l2(0.1)

))

model.add(layers.Conv2D(8, (3, 3), activation='relu', strides=2, kernel\_regularizer=regularizers.l2(0.1)

))

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu', kernel\_regularizer=regularizers.l2(0.001)

))

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

# Note on regularizer(s), copied from https://www.tensorflow.org/tutorials/keras/overfit\_and\_underfit:

# l2(0.001) means that every coefficient in the weight matrix of the layer will add 0.001 \* weight\_coefficient\_value\*\*2

# to the total loss of the network.

#model.add(layers.MaxPooling2D((2, 2)))

#model.add(layers.Dropout(0.05))

# Print summary

model.summary()

# Compile model & make some design choices

model.compile(optimizer=tf.optimizers.Adam(learning\_rate=0.0001,

beta\_1=0.9,

beta\_2=0.999,

epsilon=1e-07,

amsgrad=False,

name='Adam'

),

loss='sparse\_categorical\_crossentropy', # Capable of working with regularization

metrics=['accuracy', 'sparse\_categorical\_crossentropy'])