```
Getting started with activation function
https://github.com/PanugantiSasank123/Getting-started-with-activation-
```

Getting started with activation function
https://github.com/PanugantiSasank123/Getting-started-with-activation-function-

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf


from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from tensorflow.keras.utils import to_categorical
from keras.callbacks import Callback


from keras.datasets import fashion_mnist
(X_train, y_train), (X_val, y_val) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```python
 unique_labels = set(y_train)
 plt.figure(figsize=(12, 12))
```

```
<Figure size 1200x1200 with 0 Axes>
```

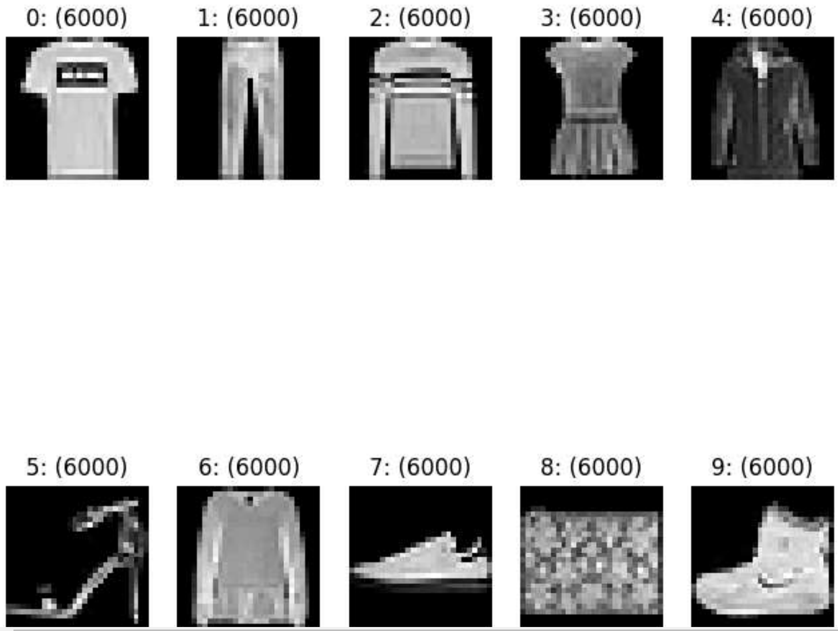
```python
y_train = np.array(y_train).flatten()


unique_labels = np.unique(y_train)


plt.figure(figsize=(8, 8))

for i, label in enumerate(unique_labels):
    indices = np.where(y_train == label)[0]
    image = X_train[indices[0]]

    plt.subplot(2, 5, i + 1)
    plt.axis('off')
    plt.title(f"{label}: ({np.sum(y_train == label)})")
    plt.imshow(image, cmap='gray')
plt.show()
```




```
print(X_val)
print(y_val)
```

```
[[[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 ...

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]

 [[0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  ...
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]
  [0 0 0 ... 0 0 0]]]
[9 2 1 ... 8 1 5]
```

```
X_train = X_train.astype('float32')/255.
X_val = X_val.astype('float32')/255.
X_val
```

```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       ...,

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]], dtype=float32)
```

```
n_classes = 10
y_train = to_categorical(y_train, n_classes)
y_val = to_categorical(y_val, n_classes)
print(y_train)
```

```
[[0. 0. 0. ... 0. 0. 1.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

```
X_train = np.reshape(X_train, (60000, 784))
X_val = np.reshape(X_val, (10000, 784))
X_train
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

```
model_sigmoid = Sequential()
model_sigmoid.add(Dense(700, input_dim=784, activation='sigmoid'))
model_sigmoid.add(Dense(700, activation='sigmoid'))
model_sigmoid.add(Dense(700, activation='sigmoid'))
model_sigmoid.add(Dense(700, activation='sigmoid'))
model_sigmoid.add(Dense(700, activation='sigmoid'))
model_sigmoid.add(Dense(350, activation='sigmoid'))
model_sigmoid.add(Dense(100, activation='sigmoid'))
model_sigmoid.add(Dense(10, activation='softmax'))
# Compile model with SGD
model_sigmoid.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argumer
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model_relu = Sequential()
model_relu.add(Dense(700, input_dim=784, activation='relu'))
model_relu.add(Dense(700, activation='relu'))
model_relu.add(Dense(700, activation='relu'))
model_relu.add(Dense(700, activation='relu'))
model_relu.add(Dense(700, activation='relu'))
model_relu.add(Dense(350, activation='relu'))
model_relu.add(Dense(100, activation='relu'))
model_relu.add(Dense(10, activation='softmax'))
# Compile model with SGD
model_relu.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

```
from keras.layers import ELU

model_elu = Sequential()
model_elu.add(Dense(700, input_dim=784))
model_elu.add(ELU())
model_elu.add(Dense(700))
model_elu.add(ELU())
model_elu.add(Dense(700))
model_elu.add(ELU())
model_elu.add(Dense(700))
model_elu.add(ELU())
model_elu.add(Dense(700))
model_elu.add(ELU())
model_elu.add(Dense(350))
model_elu.add(ELU())
model_elu.add(Dense(100))
model_elu.add(ELU())
model_elu.add(Dense(10, activation='softmax'))
# Compile model with SGD
model_elu.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

```
model_selu = Sequential()
model_selu.add(Dense(700, input_dim=784, activation='selu'))
model_selu.add(Dense(700, activation='selu'))
model_selu.add(Dense(700, activation='selu'))
model_selu.add(Dense(700, activation='selu'))
model_selu.add(Dense(700, activation='selu'))
model_selu.add(Dense(350, activation='selu'))
model_selu.add(Dense(100, activation='selu'))
model_selu.add(Dense(10, activation='softmax'))
# Compile model with SGD
model_selu.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

```
import tensorflow as tf

model_gelu = Sequential()
model_gelu.add(Dense(700, input_dim=784, activation=tf.keras.activations.gelu))
model_gelu.add(Dense(700, activation=tf.keras.activations.gelu))
model_gelu.add(Dense(700, activation=tf.keras.activations.gelu))
model_gelu.add(Dense(700, activation=tf.keras.activations.gelu))
model_gelu.add(Dense(700, activation=tf.keras.activations.gelu))
model_gelu.add(Dense(350, activation=tf.keras.activations.gelu))
model_gelu.add(Dense(100, activation=tf.keras.activations.gelu))
model_gelu.add(Dense(10, activation='softmax'))
# Compile model with SGD
model_gelu.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])



model_tanh = Sequential()
model_tanh.add(Dense(700, input_dim=784, activation='tanh'))
model_tanh.add(Dense(700, activation='tanh'))
model_tanh.add(Dense(700, activation='tanh'))
model_tanh.add(Dense(700, activation='tanh'))
model_tanh.add(Dense(700, activation='tanh'))
model_tanh.add(Dense(350, activation='tanh'))
model_tanh.add(Dense(100, activation='tanh'))
model_tanh.add(Dense(10, activation='softmax'))
# Compile model with SGD
model_tanh.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])



class history_loss(Callback):
 def on_train_begin(self, logs={}):
  self.losses = []
 def on_batch_end(self, batch, logs={}):
  batch_loss = logs.get('loss')
  self.losses.append(batch_loss)


n_epochs = 10
batch_size = 256
validation_split = 0.2
history_sigmoid = history_loss()
model_sigmoid.fit(X_train, y_train, epochs=n_epochs, batch_size=batch_size,
 callbacks=[history_sigmoid],
 validation_split=validation_split, verbose=2)
```

```
Epoch 1/10
188/188 - 21s - 112ms/step - accuracy: 0.0980 - loss: 2.3335 - val_accuracy: 0.1005 - val_loss: 2.3029
Epoch 2/10
188/188 - 20s - 106ms/step - accuracy: 0.0991 - loss: 2.3030 - val_accuracy: 0.1003 - val_loss: 2.3028
Epoch 3/10
188/188 - 19s - 103ms/step - accuracy: 0.0995 - loss: 2.3030 - val_accuracy: 0.0983 - val_loss: 2.3029
Epoch 4/10
188/188 - 22s - 118ms/step - accuracy: 0.0987 - loss: 2.3029 - val_accuracy: 0.1003 - val_loss: 2.3031
Epoch 5/10
188/188 - 19s - 102ms/step - accuracy: 0.1005 - loss: 2.3030 - val_accuracy: 0.1030 - val_loss: 2.3026
Epoch 6/10
188/188 - 22s - 118ms/step - accuracy: 0.0976 - loss: 2.3029 - val_accuracy: 0.0989 - val_loss: 2.3032
Epoch 7/10
188/188 - 41s - 218ms/step - accuracy: 0.0982 - loss: 2.3030 - val_accuracy: 0.0957 - val_loss: 2.3029
Epoch 8/10
188/188 - 19s - 99ms/step - accuracy: 0.0984 - loss: 2.3030 - val_accuracy: 0.0995 - val_loss: 2.3029
Epoch 9/10
188/188 - 22s - 118ms/step - accuracy: 0.0980 - loss: 2.3030 - val_accuracy: 0.0983 - val_loss: 2.3030
Epoch 10/10
188/188 - 19s - 100ms/step - accuracy: 0.1014 - loss: 2.3029 - val_accuracy: 0.1013 - val_loss: 2.3029
<keras.src.callbacks.history.History at 0x7fae458190c0>
```

```
 history_relu = history_loss()
 model_relu.fit(X_train, y_train, epochs=n_epochs, batch_size=batch_size,
 callbacks=[history_relu],
 validation_split=validation_split, verbose=2)
```

```
Epoch 1/10
188/188 - 22s - 115ms/step - accuracy: 0.4527 - loss: 1.9277 - val_accuracy: 0.6429 - val_loss: 1.2014
Epoch 2/10
188/188 - 19s - 101ms/step - accuracy: 0.6810 - loss: 0.9102 - val_accuracy: 0.6707 - val_loss: 0.9185
Epoch 3/10
188/188 - 21s - 114ms/step - accuracy: 0.7507 - loss: 0.7008 - val_accuracy: 0.7897 - val_loss: 0.6509
Epoch 4/10
```

```
188/188 - 19s - 100ms/step - accuracy: 0.7885 - loss: 0.6018 - val_accuracy: 0.7561 - val_loss: 0.6461
Epoch 5/10
188/188 - 20s - 108ms/step - accuracy: 0.8084 - loss: 0.5464 - val_accuracy: 0.8197 - val_loss: 0.5179
Epoch 6/10
188/188 - 21s - 110ms/step - accuracy: 0.8199 - loss: 0.5148 - val_accuracy: 0.8273 - val_loss: 0.4994
Epoch 7/10
188/188 - 21s - 109ms/step - accuracy: 0.8295 - loss: 0.4863 - val_accuracy: 0.8056 - val_loss: 0.5452
Epoch 8/10
188/188 - 21s - 109ms/step - accuracy: 0.8347 - loss: 0.4688 - val_accuracy: 0.8359 - val_loss: 0.4709
Epoch 9/10
188/188 - 22s - 115ms/step - accuracy: 0.8390 - loss: 0.4556 - val_accuracy: 0.8328 - val_loss: 0.4714
Epoch 10/10
188/188 - 19s - 103ms/step - accuracy: 0.8450 - loss: 0.4405 - val_accuracy: 0.8432 - val_loss: 0.4504
<keras.src.callbacks.history.History at 0x7fae45818a30>
```

```
history_elu = history_loss()
model_elu.fit(X_train, y_train, epochs=n_epochs, batch_size=batch_size,
              callbacks=[history_elu],
              validation_split=validation_split, verbose=2)
```

```
Epoch 1/10
188/188 - 21s - 114ms/step - accuracy: 0.7181 - loss: 0.8770 - val_accuracy: 0.7871 - val_loss: 0.6088
Epoch 2/10
188/188 - 43s - 229ms/step - accuracy: 0.8084 - loss: 0.5498 - val_accuracy: 0.8155 - val_loss: 0.5182
Epoch 3/10
188/188 - 19s - 101ms/step - accuracy: 0.8241 - loss: 0.4944 - val_accuracy: 0.8152 - val_loss: 0.5080
Epoch 4/10
188/188 - 19s - 103ms/step - accuracy: 0.8336 - loss: 0.4659 - val_accuracy: 0.8323 - val_loss: 0.4634
Epoch 5/10
188/188 - 21s - 114ms/step - accuracy: 0.8404 - loss: 0.4483 - val_accuracy: 0.8409 - val_loss: 0.4549
Epoch 6/10
188/188 - 20s - 107ms/step - accuracy: 0.8454 - loss: 0.4338 - val_accuracy: 0.8426 - val_loss: 0.4352
Epoch 7/10
188/188 - 20s - 107ms/step - accuracy: 0.8503 - loss: 0.4223 - val_accuracy: 0.8437 - val_loss: 0.4344
Epoch 8/10
188/188 - 20s - 106ms/step - accuracy: 0.8530 - loss: 0.4142 - val_accuracy: 0.8295 - val_loss: 0.4637
Epoch 9/10
188/188 - 22s - 118ms/step - accuracy: 0.8561 - loss: 0.4050 - val_accuracy: 0.8383 - val_loss: 0.4446
Epoch 10/10
188/188 - 41s - 216ms/step - accuracy: 0.8586 - loss: 0.3974 - val_accuracy: 0.8508 - val_loss: 0.4106
<keras.src.callbacks.history.History at 0x7fae4560a500>
```

```
history_selu = history_loss()
model_selu.fit(X_train, y_train, epochs=n_epochs, batch_size=batch_size,
               callbacks=[history_selu],
               validation_split=validation_split, verbose=2)
```

```
Epoch 1/10
188/188 - 21s - 114ms/step - accuracy: 0.7623 - loss: 0.7290 - val_accuracy: 0.7983 - val_loss: 0.5217
Epoch 2/10
188/188 - 20s - 106ms/step - accuracy: 0.8414 - loss: 0.4368 - val_accuracy: 0.8509 - val_loss: 0.4135
Epoch 3/10
188/188 - 19s - 102ms/step - accuracy: 0.8589 - loss: 0.3939 - val_accuracy: 0.8577 - val_loss: 0.3882
Epoch 4/10
188/188 - 22s - 119ms/step - accuracy: 0.8661 - loss: 0.3694 - val_accuracy: 0.8584 - val_loss: 0.3915
Epoch 5/10
188/188 - 19s - 102ms/step - accuracy: 0.8725 - loss: 0.3530 - val_accuracy: 0.8638 - val_loss: 0.3736
Epoch 6/10
188/188 - 22s - 118ms/step - accuracy: 0.8803 - loss: 0.3340 - val_accuracy: 0.8707 - val_loss: 0.3593
Epoch 7/10
188/188 - 19s - 100ms/step - accuracy: 0.8839 - loss: 0.3208 - val_accuracy: 0.8704 - val_loss: 0.3598
Epoch 8/10
188/188 - 23s - 120ms/step - accuracy: 0.8885 - loss: 0.3102 - val_accuracy: 0.8619 - val_loss: 0.3779
Epoch 9/10
188/188 - 41s - 219ms/step - accuracy: 0.8926 - loss: 0.2996 - val_accuracy: 0.8770 - val_loss: 0.3362
Epoch 10/10
188/188 - 19s - 98ms/step - accuracy: 0.8945 - loss: 0.2902 - val_accuracy: 0.8669 - val_loss: 0.3690
<keras.src.callbacks.history.History at 0x7fae2e6974c0>
```

```
history_gelu = history_loss()
model_gelu.fit(X_train, y_train, epochs=n_epochs, batch_size=batch_size,
               callbacks=[history_gelu],
               validation_split=validation_split, verbose=2)
```

```
Epoch 1/10
188/188 - 24s - 129ms/step - accuracy: 0.1389 - loss: 2.2975 - val_accuracy: 0.1968 - val_loss: 2.2913
Epoch 2/10
```

```
188/188 - 41s - 216ms/step - accuracy: 0.2224 - loss: 2.2834 - val_accuracy: 0.2294 - val_loss: 2.2717
Epoch 3/10
188/188 - 42s - 223ms/step - accuracy: 0.1971 - loss: 2.2430 - val_accuracy: 0.1600 - val_loss: 2.1754
Epoch 4/10
188/188 - 40s - 211ms/step - accuracy: 0.2755 - loss: 2.0479 - val_accuracy: 0.4033 - val_loss: 1.8085
Epoch 5/10
188/188 - 41s - 217ms/step - accuracy: 0.4567 - loss: 1.4398 - val_accuracy: 0.5470 - val_loss: 1.1823
Epoch 6/10
188/188 - 21s - 110ms/step - accuracy: 0.6163 - loss: 1.0373 - val_accuracy: 0.6358 - val_loss: 0.8928
Epoch 7/10
188/188 - 23s - 123ms/step - accuracy: 0.6943 - loss: 0.8445 - val_accuracy: 0.7096 - val_loss: 0.8078
Epoch 8/10
188/188 - 40s - 213ms/step - accuracy: 0.7361 - loss: 0.7270 - val_accuracy: 0.7503 - val_loss: 0.7070
Epoch 9/10
188/188 - 41s - 219ms/step - accuracy: 0.7533 - loss: 0.6737 - val_accuracy: 0.7575 - val_loss: 0.6627
Epoch 10/10
188/188 - 41s - 217ms/step - accuracy: 0.7670 - loss: 0.6289 - val_accuracy: 0.7782 - val_loss: 0.6108
<keras.src.callbacks.history.History at 0x7fae2ee3e2c0>
```

```
history_tanh = history_loss()
model_tanh.fit(X_train, y_train, epochs=n_epochs, batch_size=batch_size,
               callbacks=[history_tanh],
               validation_split=validation_split, verbose=2)
```

```
Epoch 1/10
188/188 - 23s - 124ms/step - accuracy: 0.7198 - loss: 0.8843 - val_accuracy: 0.7910 - val_loss: 0.6220
Epoch 2/10
188/188 - 39s - 207ms/step - accuracy: 0.8087 - loss: 0.5662 - val_accuracy: 0.8148 - val_loss: 0.5331
Epoch 3/10
188/188 - 20s - 108ms/step - accuracy: 0.8260 - loss: 0.5020 - val_accuracy: 0.8290 - val_loss: 0.4890
Epoch 4/10
188/188 - 19s - 103ms/step - accuracy: 0.8352 - loss: 0.4707 - val_accuracy: 0.8313 - val_loss: 0.4687
Epoch 5/10
188/188 - 22s - 118ms/step - accuracy: 0.8428 - loss: 0.4488 - val_accuracy: 0.8381 - val_loss: 0.4509
Epoch 6/10
188/188 - 40s - 211ms/step - accuracy: 0.8474 - loss: 0.4344 - val_accuracy: 0.8453 - val_loss: 0.4363
Epoch 7/10
188/188 - 19s - 100ms/step - accuracy: 0.8513 - loss: 0.4230 - val_accuracy: 0.8381 - val_loss: 0.4520
Epoch 8/10
188/188 - 22s - 117ms/step - accuracy: 0.8540 - loss: 0.4130 - val_accuracy: 0.8496 - val_loss: 0.4176
Epoch 9/10
188/188 - 19s - 101ms/step - accuracy: 0.8568 - loss: 0.4051 - val_accuracy: 0.8460 - val_loss: 0.4255
Epoch 10/10
188/188 - 19s - 100ms/step - accuracy: 0.8602 - loss: 0.3961 - val_accuracy: 0.8474 - val_loss: 0.4213
<keras.src.callbacks.history.History at 0x7fae2ec6ad10>
```

```
np.arange(len(history_sigmoid.losses))
print(history_sigmoid.losses)
```

```
[2.763857841491699, 2.7381319999694824, 2.6699044704437256, 2.6578569412231445, 2.659571886062622, 2.63161039352417, 2.6143124103546143,
```

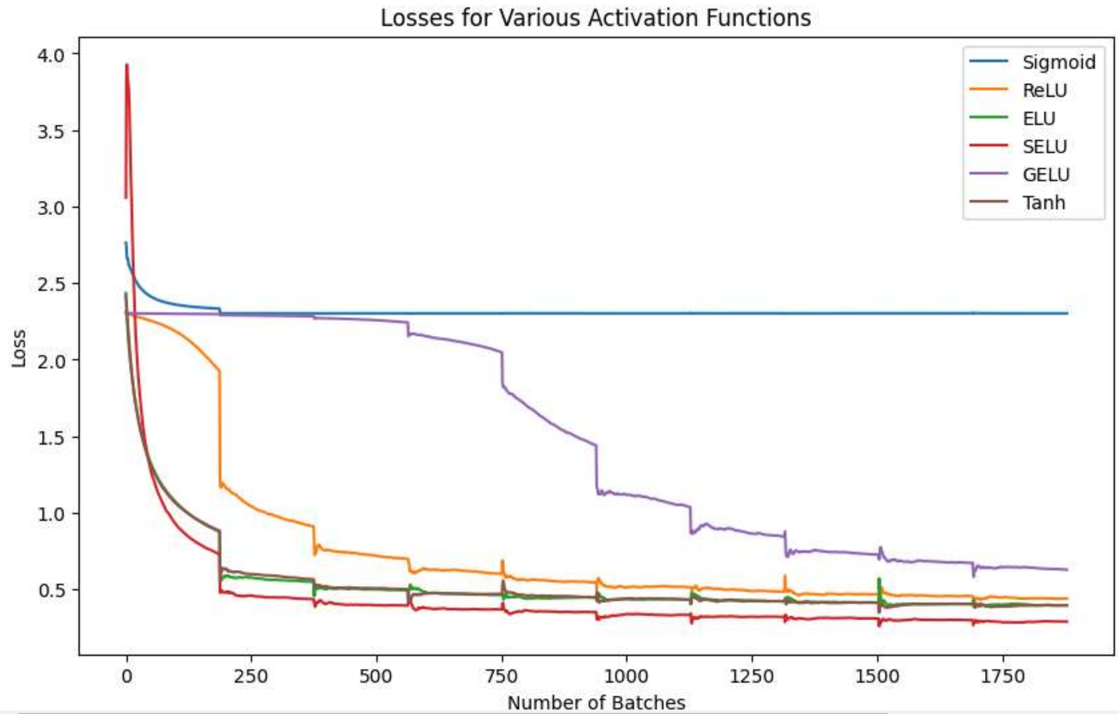
```
# Plot loss curves for all activation functions
plt.figure(figsize=(10, 6))

# Plotting loss for each activation function
plt.plot(np.arange(len(history_sigmoid.losses)), history_sigmoid.losses, label='Sigmoid')
plt.plot(np.arange(len(history_relu.losses)), history_relu.losses, label='ReLU')
plt.plot(np.arange(len(history_elu.losses)), history_elu.losses, label='ELU')
plt.plot(np.arange(len(history_selu.losses)), history_selu.losses, label='SELU')
plt.plot(np.arange(len(history_gelu.losses)), history_gelu.losses, label='GELU')
plt.plot(np.arange(len(history_tanh.losses)), history_tanh.losses, label='Tanh')

# Add titles and labels
plt.title('Losses for Various Activation Functions')
plt.xlabel('Number of Batches')
plt.ylabel('Loss')

# Add a legend
plt.legend(loc='best')

# Show the plot
plt.show()
```




```
w_sigmoid = []
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.