In [1]:
```
import tensorflow as tf
from tensorflow.keras import layers,models
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train,y_train),(x_test,y_test) = fashion_mnist.load_data()
```

In [3]:
```
x_train,x_test = x_train/255.0,x_test/255.0
```

In [6]:
```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28,28)))
model.add(layers.Dense(128,activation='relu'))
model.add(layers.Dense(10,activation='softmax'))
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metri
```

In [7]:
```
model.fit(x_train,y_train,epochs=10,batch_size=32,validation_data=(x_test,y_
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7827 - loss: 0.63
70 - val_accuracy: 0.8387 - val_loss: 0.4585
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8574 - loss: 0.38
93 - val_accuracy: 0.8547 - val_loss: 0.4060
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8755 - loss: 0.33
94 - val_accuracy: 0.8680 - val_loss: 0.3724
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.8841 - loss: 0.3
131 - val_accuracy: 0.8720 - val_loss: 0.3632
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8885 - loss: 0.30
37 - val_accuracy: 0.8757 - val_loss: 0.3511
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8944 - loss: 0.27
85 - val_accuracy: 0.8758 - val_loss: 0.3443
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8992 - loss: 0.26
87 - val_accuracy: 0.8815 - val_loss: 0.3452
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9051 - loss: 0.25
81 - val_accuracy: 0.8846 - val_loss: 0.3292
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9055 - loss: 0.24
96 - val_accuracy: 0.8789 - val_loss: 0.3399
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9092 - loss: 0.24
15 - val_accuracy: 0.8834 - val_loss: 0.3251
```

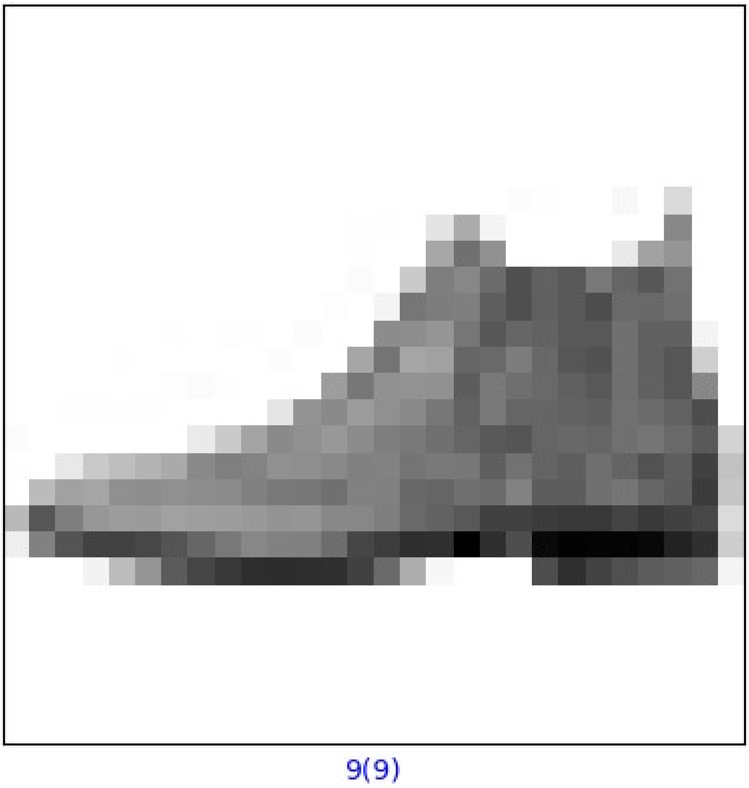
Out[7]:
```
<keras.src.callbacks.history.History at 0x216ad49cad0>
```

In [8]:
```
test_loss,test_acc = model.evaluate(x_test,y_test)
print(f'Test accuracy:{test_acc}')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8816 - loss: 0.3259
Test accuracy:0.883400022983551
```

In [10]:
```
predictions = model.predict(x_test)

def plot_image(i,predictions_array,true_label,img):
    predictions_array,true_label,img = predictions_array[i],true_label[i],i
    plt.grid(True)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(img,cmap=plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    color = 'blue' if predicted_label == true_label else 'red'
    plt.xlabel(f'{predicted_label}({true_label})',color=color)

plot_image(0,predictions,y_test,x_test)
plt.show()
```

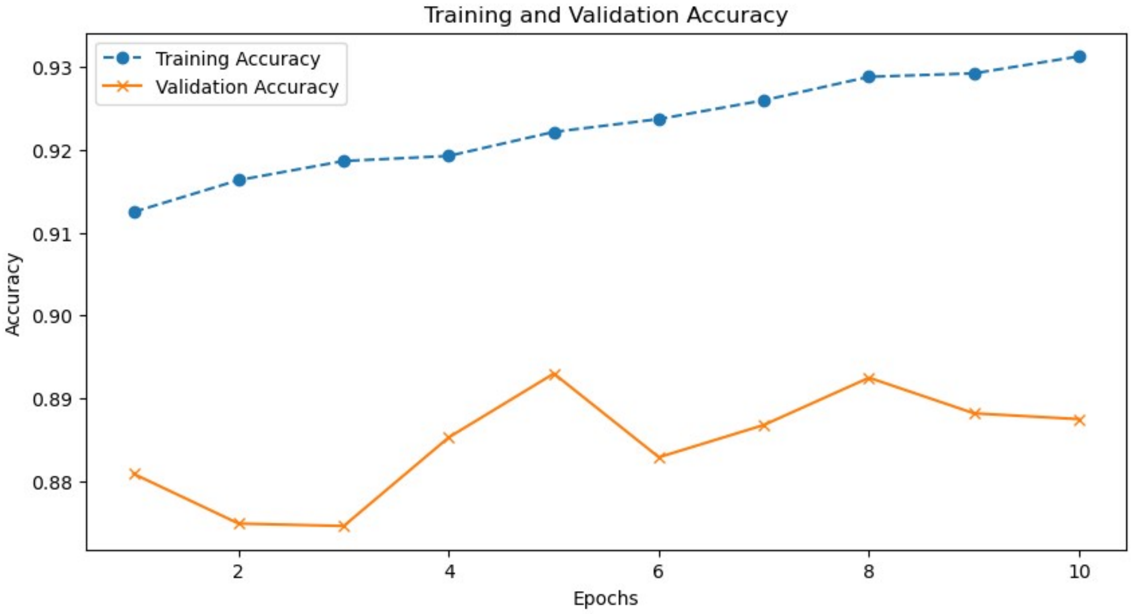
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```



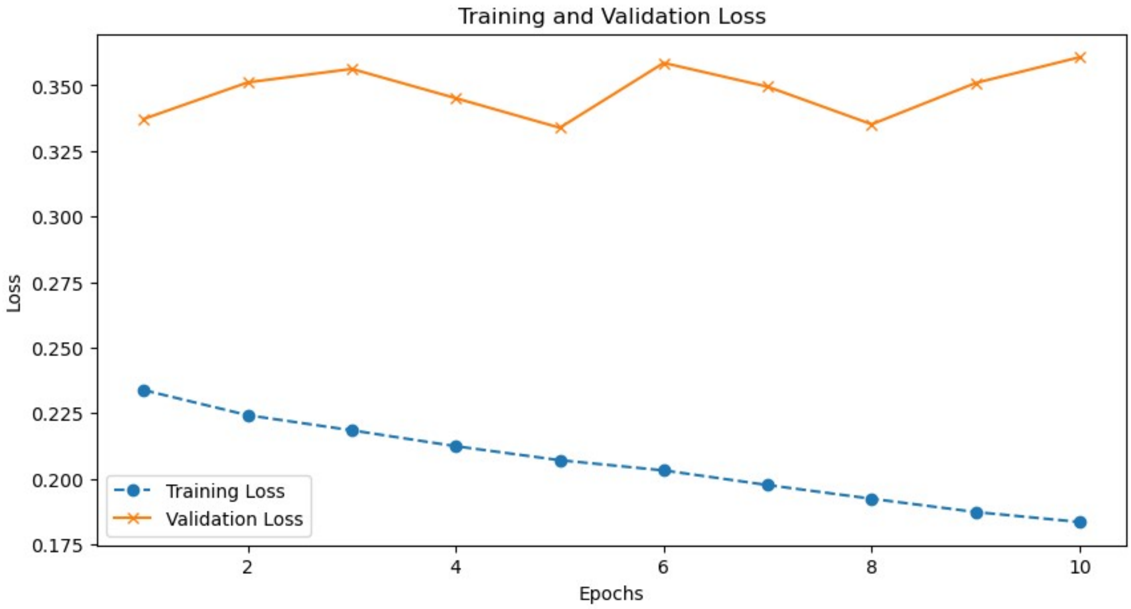
```
In [11]: history = model.fit(x_train,y_train,epochs=10,batch_size=32,validation_data
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9124 - loss: 0.23
20 - val_accuracy: 0.8809 - val_loss: 0.3372
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9193 - loss: 0.22
02 - val_accuracy: 0.8749 - val_loss: 0.3512
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9198 - loss: 0.21
59 - val_accuracy: 0.8746 - val_loss: 0.3563
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9190 - loss: 0.21
43 - val_accuracy: 0.8853 - val_loss: 0.3451
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9226 - loss: 0.20
52 - val_accuracy: 0.8930 - val_loss: 0.3339
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9239 - loss: 0.20
34 - val_accuracy: 0.8829 - val_loss: 0.3586
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9267 - loss: 0.19
34 - val_accuracy: 0.8868 - val_loss: 0.3495
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9292 - loss: 0.19
03 - val_accuracy: 0.8925 - val_loss: 0.3352
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9296 - loss: 0.18
37 - val_accuracy: 0.8882 - val_loss: 0.3509
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9332 - loss: 0.18
11 - val_accuracy: 0.8875 - val_loss: 0.3608
```

In [12]:
```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1,len(acc)+1)

plt.figure(figsize=(10,5))
plt.plot(epochs,acc,label="Training Accuracy",linestyle="--",marker='o')
plt.plot(epochs,val_acc,label="Validation Accuracy",linestyle='-',marker='x
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [13]:
```
plt.figure(figsize=(10,5))
plt.plot(epochs,loss,label='Training Loss',linestyle='--',marker='o')
plt.plot(epochs,val_loss,label='Validation Loss',linestyle='-',marker='x')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



In [1]:
```
%pip install -q -U keras-tuner
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
%pip install -q -U keras-tuner
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [25]:
```
#Fine tuning Hyperparameters
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers,models
from sklearn.model_selection import RandomizedSearchCV
from scikeras.wrappers import KerasClassifier
from tensorflow.keras.datasets import fashion_mnist
```

In [26]:
```
(X_train,y_train),(X_test,y_test) = fashion_mnist.load_data()

#Normalize
X_train = X_train/255.0
X_test = X_test/255.0

#Flatten
X_train = X_train.reshape(X_train.shape[0],28*28)
X_test = X_test.reshape(X_test.shape[0],28*28)
```

In [27]:

```
#Create ANN
def create_ann(optimizer='adam',init_mode='glorot_uniform',activation='relu
    model = models.Sequential()
    #Input Layer
    model.add(layers.Dense(neurons,activation=activation,kernel_initializer

    #Hidden Layer
    model.add(layers.Dense(neurons,activation=activation))
    model.add(layers.Dropout(dropout_rate))

    #Output Layer
    model.add(layers.Dense(10,activation='softmax'))

    #Compile the model
    model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy

    return model
```

In [28]:

```
#Define Hyperparametr search space
param_grid = {
    'batch_size':[32,64,128],
    'epochs':[10,20],
    'optimizer':['adam','sgd'],
    'init_mode':['uniform','glorot_uniform'],
    'activation':['relu','tanh'],
    'dropout_rate':[0.2,0.3],
    'neurons':[64,128,256]
}
```

In [34]:
```
model = KerasClassifier(model=create_ann,verbose=0,neurons=256,init_mode='g

random_search = RandomizedSearchCV(estimator=model,param_distributions=para

random_search_result = random_search.fit(X_train,y_train) ##use keras tuner
```

```
Fitting 3 folds for each of 5 candidates, totalling 15 fits
```

```
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [23]:

```python
import numpy as np
def unitStep(v):
    if v>=0:
        return 1
    else :
        return 0
def perceptronModel(x,w,b):
    v = np.dot(w,x)+b
    y = unitStep(v)
    return y

def NOT_logicFunction(x):
    wNOT = -1
    bNOT = 0.5
    return perceptronModel(x,wNOT,bNOT)

def AND_logicFunction(x):
    w = np.array([1,1])
    bAND = -1.5
    return perceptronModel(x,w,bAND)

def OR_logicFunction(x):
    w = np.array([1,1])
    bOR = -0.5
    return perceptronModel(x,w,bOR)

def XOR_logicFunction(x):
    y1 = AND_logicFunction(x)
    y2 = OR_logicFunction(x)
    y3 = NOT_logicFunction(y1)
    final_x = np.array([y2,y3])
    finalOutput = AND_logicFunction(final_x)
    return finalOutput
```

In [24]:
```
test1 = np.array([0,1])
test2 = np.array([1,1])
test3 = np.array([0,0])
test4 = np.array([1,0])

print('XOR({},{})={}'.format(0,1,XOR_logicFunction(test1)))
print('XOR({},{})={}'.format(1,1,XOR_logicFunction(test2)))
print('XOR({},{})={}'.format(0,0,XOR_logicFunction(test3)))
print('XOR({},{})={}'.format(1,0,XOR_logicFunction(test4)))
```

```
XOR(0,1)=1
XOR(1,1)=0
XOR(0,0)=0
XOR(1,0)=1
```

In [25]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import SGD
```

In [26]:
```
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

model = Sequential()

model.add(Dense(2,input_dim=2,activation='relu',name='hidden_layer'))
model.add(Dense(1,activation='sigmoid',name='output_layer'))
optimizer = SGD(learning_rate=0.1)
model.compile(optimizer=optimizer,loss='binary_crossentropy',metrics=['accu

model.summary()
```

```
Model: "sequential_1"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━
┃ Layer (type)                    ┃ Output Shape           ┃       Param #
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━
│ hidden_layer (Dense)            │ (None, 2)              │             6
├─────────────────────────────────┼────────────────────────┼───────────────
│ output_layer (Dense)            │ (None, 1)              │             3
└─────────────────────────────────┴────────────────────────┴───────────────

 Total params: 9 (36.00 B)

 Trainable params: 9 (36.00 B)

 Non-trainable params: 0 (0.00 B)
```

In [38]:

```
class WeightsTracker(tf.keras.callbacks.Callback):
    def on_epoch_end(self,epoch,logs=None):

        hidden_weights,hidden_biases = self.model.get_layer('hidden_layer')
        output_weights,output_biases = self.model.get_layer('output_layer')

        print(f"\nEpoch {epoch+1}:")
        print("Hidden Layer Weights:\n",hidden_weights)
        print("Hidden Layer Biases:\n",hidden_biases)
        print("Output Layer Weights:\n",output_weights)
        print("Output Layer Biases:\n",output_biases)

model.fit(X,y,epochs=20,verbose=0,callbacks=[WeightsTracker()])
print('\nPredictions after training:')
print(model.predict(X))
```

```
Epoch 1:
Hidden Layer Weights:
 [[ 0.62695116  0.1974517 ]
 [-0.29097292  0.98797554]]
Hidden Layer Biases:
 [-0.0256984  -0.19979142]
Output Layer Weights:
 [[ 0.58515227]
 [-0.49714226]]
Output Layer Biases:
 [0.0696606]

Epoch 2:
Hidden Layer Weights:
 [[ 0.6263013   0.20292756]
 [-0.29741818  0.9862446 ]]
Hidden Layer Biases:
 [-0.02634829 -0.20152232]
```

In [49]:
```
import numpy as np
from sklearn.linear_model import Perceptron
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

class MyPerceptron:
    def __init__(self,learning_rate=0.1,n_iterations=100):
        self.learning_rate = learning_rate
        self.n_iterations = n_iterations
        self.weights = None
        self.bias = None



    def fit(self,X,y):
        n_samples,n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0
        for _ in range(self.n_iterations):
            for i in range(n_samples):
                linear_output = np.dot(X[i],self.weights) + self.bias()
                y_predicted = self.activation_function(linear_output)

                update = self.learning_rate * (y[i]-y_predicted)
                self.weights += update * X[i]
                print(self.weights)

    def predict(self,X):
        linear_output = np.dot(X,self.weights) + self.bias
        y_predicted = self.activation_function(linear_output)
        return y_predicted

    def activation_function(self,X):
        return np.where(X>=0,1,0)

my_perceptron = MyPerceptron()
my_perceptron.fit(X,y)
print(my_perceptron.predict(X))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[49], line 37
     34         return np.where(X>=0,1,0)
     36 my_perceptron = MyPerceptron()
---> 37 my_perceptron.fit(X,y)
     38 print(my_perceptron.predict(X))

Cell In[49], line 21, in MyPerceptron.fit(self, X, y)
     19 for _ in range(self.n_iterations):
     20     for i in range(n_samples):
---> 21         linear_output = np.dot(X[i],self.weights) + self.bias()
     22         y_predicted = self.activation_function(linear_output)
     24         update = self.learning_rate * (y[i]-y_predicted)

TypeError: 'int' object is not callable
```

In [1]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras_tuner import RandomSearch
```

In [14]:

```
def build_model(hp):
    model = Sequential()

    hp_neurons = hp.Int(128,min_value=10,max_value=100,step=10)
    model.add(Dense(units=hp_neurons,activation=hp.Choice('activation',['ta
    model.add(Dense(1,activation='sigmoid'))

    model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['acc
    return model
```

In [15]:

```
tuner = RandomSearch(
    build_model,
    objective='val_accuracy',
    max_trials=10,
    executions_per_trial=1,
    project_name='ann_tuning'
)
```

```
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [21]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data = load_breast_cancer()
X,y = data.data,data.target
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_s

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

tuner.search(X_train,y_train,epochs=50,validation_split=0.2)
```

```
Trial 2 Complete [00h 00m 00s]

Best val_accuracy So Far: None
Total elapsed time: 00h 00m 02s

Search: Running Trial #3

Value             |Best Value So Far |Hyperparameter
20                |100               |128
relu              |relu              |activation

Epoch 1/50

Traceback (most recent call last):
  File "C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\kera
s_tuner\src\engine\base_tuner.py", line 274, in _try_run_and_update_trial
    self._run_and_update_trial(trial, *fit_args, **fit_kwargs)
  File "C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\kera
s_tuner\src\engine\base_tuner.py", line 239, in _run_and_update_trial
    results = self.run_trial(trial, *fit_args, **fit_kwargs)
```

In [2]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropo
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.utils import to_categorical
from functools import partial


(x_train,y_train),(x_test,y_test) = fashion_mnist.load_data()

x_train = x_train.reshape(-1,28,28,1).astype('float32')/255.0
x_test = x_test.reshape(-1,28,28,1).astype('float32')/255.0

y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)

ConvLayer = partial(Conv2D,kernel_size=(3,3),activation='relu',padding='sam

model = Sequential([
    ConvLayer(32,input_shape=(28,28,1)),
    MaxPooling2D(pool_size=(2,2)),
    ConvLayer(64),
    MaxPooling2D(pool_size=(2,2)),
    ConvLayer(128),
    MaxPooling2D(pool_size=(2,2)),
    Flatten(),
    Dense(128,activation='relu'),
    Dropout(0.5),
    Dense(10,activation='softmax')
])
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['ac
model.fit(x_train,y_train,epochs=10,batch_size=32,validation_split=0.2)

test_loss,test_accuracy = model.evaluate(x_test,y_test)
print(f"Test Loss:{test_loss},Test Accuracy:{test_accuracy}")
```

```
Epoch 1/10

C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_sh
ape`/`input_dim` argument to a layer. When using Sequential models, prefer
using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 13ms/step - accuracy: 0.6983 - loss: 0.
8194 - val_accuracy: 0.8684 - val_loss: 0.3498
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 20s 13ms/step - accuracy: 0.8656 - loss: 0.
3771 - val_accuracy: 0.8837 - val_loss: 0.3024
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 20s 13ms/step - accuracy: 0.8883 - loss: 0.
3125 - val_accuracy: 0.9007 - val_loss: 0.2765
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 16s 10ms/step - accuracy: 0.9020 - loss: 0.
2674 - val_accuracy: 0.9144 - val_loss: 0.2349
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 14s 9ms/step - accuracy: 0.9136 - loss: 0.2
416 - val_accuracy: 0.9041 - val_loss: 0.2551
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 17s 11ms/step - accuracy: 0.9206 - loss: 0.
2155 - val_accuracy: 0.9157 - val_loss: 0.2395
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 22s 12ms/step - accuracy: 0.9270 - loss: 0.
1976 - val_accuracy: 0.9155 - val_loss: 0.2319
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 13ms/step - accuracy: 0.9326 - loss: 0.
1832 - val_accuracy: 0.9167 - val_loss: 0.2337
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 13ms/step - accuracy: 0.9390 - loss: 0.
1717 - val_accuracy: 0.9200 - val_loss: 0.2299
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 19s 13ms/step - accuracy: 0.9437 - loss: 0.
1489 - val_accuracy: 0.9190 - val_loss: 0.2432
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9107 - loss: 0.2848
Test Loss:0.27270758152008057,Test Accuracy:0.9135000109672546
```

In [3]:
```
model.summary()
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━
┃ Layer (type)                    ┃ Output Shape           ┃       Param #
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━
│ conv2d (Conv2D)                 │ (None, 28, 28, 32)     │           320
├─────────────────────────────────┼────────────────────────┼──────────────
│ max_pooling2d (MaxPooling2D)    │ (None, 14, 14, 32)     │             0
├─────────────────────────────────┼────────────────────────┼──────────────
│ conv2d_1 (Conv2D)               │ (None, 14, 14, 64)     │        18,496
├─────────────────────────────────┼────────────────────────┼──────────────
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 7, 7, 64)       │             0
├─────────────────────────────────┼────────────────────────┼──────────────
│ conv2d_2 (Conv2D)               │ (None, 7, 7, 128)      │        73,856
├─────────────────────────────────┼────────────────────────┼──────────────
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 3, 3, 128)      │             0
├─────────────────────────────────┼────────────────────────┼──────────────
│ flatten (Flatten)               │ (None, 1152)           │             0
├─────────────────────────────────┼────────────────────────┼──────────────
│ dense (Dense)                   │ (None, 128)            │       147,584
├─────────────────────────────────┼────────────────────────┼──────────────
│ dropout (Dropout)               │ (None, 128)            │             0
├─────────────────────────────────┼────────────────────────┼──────────────
│ dense_1 (Dense)                 │ (None, 10)             │         1,290
└─────────────────────────────────┴────────────────────────┴──────────────

 Total params: 724,640 (2.76 MB)

 Trainable params: 241,546 (943.54 KB)

 Non-trainable params: 0 (0.00 B)

 Optimizer params: 483,094 (1.84 MB)
```

In [2]:

```
from tensorflow.keras import layers,models,Input

def conv_block(x,filters,kernel_size=3,strides=1,activation=True):
    x = layers.Conv2D(filters,kernel_size=kernel_size,strides=strides,paddi
    x = layers.BatchNormalization()(x)
    if activation:
        x = layers.ReLU()(x)
    return x

def residual_block(x,filters,downsample=False):
    shortcut = x
    strides = 2 if downsample else 1
    #First convolution layer with activation true and strides
    x = conv_block(x,filters,strides = strides)
    #Second convolution layer without activation
    x = conv_block(x,filters,activation=False)

    #Adjust shortcut if downsampling or if dimensions change
    if downsample or x.shape[-1] != shortcut.shape[-1]:
        shortcut = conv_block(shortcut,filters,kernel_size=1,strides=stride

    x = layers.add([x,shortcut])
    x = layers.ReLU()(x)
    return x

def resnet34(input_shape=(224,224,3),num_classes=1000):
    inputs = Input(shape=input_shape)

    x = conv_block(inputs,64,kernel_size=7,strides=2)
    x = layers.MaxPooling2D(pool_size=3,strides=2,padding='same')(x)

    #Residual blocks
    x = residual_block(x,64)
    x = residual_block(x,64)
    x = residual_block(x,64)

    x = residual_block(x,128,downsample=True)
    x = residual_block(x,128)
    x = residual_block(x,128)
    x = residual_block(x,128)

    x = residual_block(x,256,downsample=True)
    x = residual_block(x,256)
    x = residual_block(x,256)
    x = residual_block(x,256)
    x = residual_block(x,256)

    x = residual_block(x,512,downsample=True)
    x = residual_block(x,512)
    x = residual_block(x,512)

    x= layers.GlobalAveragePooling2D()(x)
    outputs = layers.Dense(num_classes,activation='softmax')(x)

    model = models.Model(inputs,outputs)
    return model

model = resnet34(input_shape=(224,224,3),num_classes=1000)
model.summary()
```

```
Model: "functional"

┃ Layer (type)             ┃ Output Shape          ┃      Param # ┃ Connected to          ┃
│ input_layer              │ (None, 224, 224,      │            0 │ -                     │
│ (InputLayer)             │ 3)                    │              │                       │
│ conv2d (Conv2D)          │ (None, 112, 112,      │        9,472 │ input_layer[0][0]     │
│                          │ 64)                   │              │                       │
│ batch_normalization      │ (None, 112, 112,      │          256 │ conv2d[0][0]          │
│ (BatchNormalizatio…      │ 64)                   │              │                       │
│ re_lu (ReLU)             │ (None, 112, 112,      │            0 │ batch_normalizat…     │
│                          │ 64)                   │              │                       │
│ max_pooling2d            │ (None, 56, 56,        │            0 │ re_lu[0][0]           │
│ (MaxPooling2D)           │ 64)                   │              │                       │
│ conv2d_1 (Conv2D)        │ (None, 56, 56,        │       36,928 │ max_pooling2d[0]…     │
```

In [3]:

```python
from tensorflow.keras.applications import ResNet50
from tensorflow.keras import layers,models

def build_resnet_model(input_shape=(224,224,3),num_classes=1000):
    base_model = ResNet50(weights='imagenet',include_top=False,input_shape=

    base_model.trainable = False #freeze the model for transfer learning

    x = layers.GlobalAveragePooling2D()(base_model.output)
    x = layers.Dense(512,activation='relu')(x)
    outputs = layers.Dense(num_classes,activation='softmax')(x)

    model = models.Model(inputs=base_model.input,outputs=outputs)
    return model

model = build_resnet_model(input_shape=(224,224,3),num_classes=1000)
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appl
ications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5 (http
s://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_w
eights_tf_dim_ordering_tf_kernels_notop.h5)
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 7s 0us/step

Model: "functional_1"

┃ Layer (type)             ┃ Output Shape          ┃      Param # ┃ Connected to          ┃
│ input_layer_1            │ (None, 224, 224,      │            0 │ -                     │
│ (InputLayer)             │ 3)                    │              │                       │
│ conv1_pad                │ (None, 230, 230,      │            0 │ input_layer_1[0]…     │
│ (ZeroPadding2D)          │ 3)                    │              │                       │
│ conv1_conv (Conv2D)      │ (None, 112, 112,      │        9,472 │ conv1_pad[0][0]       │
│                          │ 64)                   │              │                       │
│ conv1_bn                 │ (None, 112, 112,      │          256 │ conv1_conv[0][0]      │
```

In [4]:
```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras import layers,models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(x_train,y_train),(x_test,y_test) = mnist.load_data()
x_train = tf.image.resize_with_pad(tf.expand_dims(x_train,-1),32,32).numpy(
x_test = tf.image.resize_with_pad(tf.expand_dims(x_test,-1),32,32).numpy()/
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)

base_model = ResNet50(weights='imagenet',include_top=False,input_shape=(32,

base_model.trainable = False #freezing the model

model = models.Sequential([
    layers.Input(shape=(32,32,1)),
    layers.Conv2D(3,(3,3),padding='same'),
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128,activation='relu'),
    layers.Dense(10,activation='softmax')
])

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['ac
model.fit(x_train,y_train,epochs=5,batch_size=64,validation_data=(x_test,y_
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-data
sets/mnist.npz)
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Epoch 1/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 252s 265ms/step - accuracy: 0.6072 - loss: 1.
2014 - val_accuracy: 0.8839 - val_loss: 0.3676
Epoch 2/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 249s 266ms/step - accuracy: 0.8871 - loss: 0.
3596 - val_accuracy: 0.9147 - val_loss: 0.2722
Epoch 3/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 240s 256ms/step - accuracy: 0.9128 - loss: 0.
2760 - val_accuracy: 0.9260 - val_loss: 0.2303
Epoch 4/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 246s 262ms/step - accuracy: 0.9244 - loss: 0.
2387 - val_accuracy: 0.9283 - val_loss: 0.2228
Epoch 5/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 242s 258ms/step - accuracy: 0.9247 - loss: 0.
2286 - val_accuracy: 0.9396 - val_loss: 0.1925

Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━
┃ Layer (type)                    ┃ Output Shape           ┃       Param #
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━
│ conv2d_34 (Conv2D)              │ (None, 32, 32, 3)      │            30
├─────────────────────────────────┼────────────────────────┼──────────────
│ resnet50 (Functional)           │ (None, 1, 1, 2048)     │    23,587,712
├─────────────────────────────────┼────────────────────────┼──────────────
│ global_average_pooling2d_2      │ (None, 2048)           │             0
│ (GlobalAveragePooling2D)        │                        │
├─────────────────────────────────┼────────────────────────┼──────────────
│ dense_3 (Dense)                 │ (None, 128)            │       262,272
├─────────────────────────────────┼────────────────────────┼──────────────
│ dense_4 (Dense)                 │ (None, 10)             │         1,290
└─────────────────────────────────┴────────────────────────┴──────────────

 Total params: 24,378,490 (93.00 MB)

 Trainable params: 263,592 (1.01 MB)

 Non-trainable params: 23,587,712 (89.98 MB)

 Optimizer params: 527,186 (2.01 MB)
```

```
In [ ]:
```