Experiment No.3

Name :Sahil Tike
Class : BE-(B3)
Roll No : B212066

In [ ]:
```
train_df =pd.read_csv("../input/fashionmnist/fashion-mnist_train.csv")
test_df =pd.read_csv("../input/fashionmnist/fashion-mnist_test.csv")
```

In [3]:
```
train_df.head(30)
```

Out[3]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel7
0	2	0	0	0	0	0	0	0	0	0	...	0	
1	9	0	0	0	0	0	0	0	0	0	...	0	
2	6	0	0	0	0	0	0	0	5	0	...	0	
3	0	0	0	0	1	2	0	0	0	0	...	3	
4	3	0	0	0	0	0	0	0	0	0	...	0	
5	4	0	0	0	5	4	5	5	3	5	...	7	
6	4	0	0	0	0	0	0	0	0	0	...	14	
7	5	0	0	0	0	0	0	0	0	0	...	0	
8	4	0	0	0	0	0	0	3	2	0	...	1	
9	8	0	0	0	0	0	0	0	0	0	...	203	2
10	0	0	0	0	0	1	0	0	0	0	...	164	1
11	8	0	0	0	0	0	0	0	0	0	...	9	
12	9	0	0	0	0	0	0	0	0	0	...	0	
13	0	0	0	0	0	0	0	0	0	0	...	0	
14	2	0	0	0	0	1	1	0	0	0	...	0	
15	2	0	0	0	0	0	0	0	0	16	...	0	
16	9	0	0	0	0	0	0	0	0	0	...	0	
17	3	0	0	0	0	0	0	0	0	0	...	101	
18	3	0	0	0	0	0	0	0	0	0	...	0	
19	3	0	0	0	0	0	0	0	0	0	...	0	
20	8	0	0	0	0	0	0	0	0	0	...	80	
21	7	0	0	0	0	0	0	0	0	0	...	0	
22	4	0	0	0	0	0	0	0	0	0	...	69	
23	4	0	0	0	0	0	1	0	1	0	...	0	
24	0	0	0	0	0	0	0	0	0	40	...	125	
25	4	0	0	0	0	2	0	0	0	1	...	4	
26	4	0	0	0	0	0	0	0	1	0	...	153	
27	8	0	0	0	0	0	0	0	1	0	...	15	
28	7	0	0	0	0	0	0	0	0	0	...	0	
29	1	0	0	0	0	0	0	0	0	0	...	0	

30 rows × 785 columns

In [4]:
```
train=train_df
test=test_df
```

In [5]:
```
train_df.label.unique
```

Out[5]:
```
<bound method Series.unique of 0         2
1         9
2         6
3         0
4         3
         ..
59995     9
59996     1
59997     8
59998     8
59999     7
Name: label, Length: 60000, dtype:  int64>
```

In [6]:
```
# Remove rows with missing target, separate target from predictors

train.dropna(axis=0, subset=['label'], inplace=True)
y_train=train.label
train.drop(['label'], axis=1, inplace=True)
```

In [7]:
```
test.columns
```

Out[7]:
```
Index(['label', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5', 'pixel6',
       'pixel7', 'pixel8', 'pixel9',
       ...
       'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779', 'pixel78
0',
       'pixel781', 'pixel782', 'pixel783', 'pixel784'],
      dtype='object', length=785)
```

In [8]:
```
# Remove rows with missing target, separate target from predictors
test
test.dropna(axis=0, subset=['label'], inplace=True)
y_valid=test.label
test.drop(['label'], axis=1, inplace=True)
```

In [9]:
```
test.shape
```

Out[9]:
```
(10000, 784)
```

In [10]:
```
train.shape
```

Out[10]:
```
(60000, 784)
```

In [11]:
```
import tensorflow.keras as keras
# Separate our our image vectors
x_train = train.values
x_valid = test.values

# Turn our scalar targets into binary categories
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_valid = keras.utils.to_categorical(y_valid, num_classes)

# Normalize our image data
x_train = x_train / 255
x_valid = x_valid / 255

# Reshape the image data for the convolutional network
x_train = x_train.reshape(-1,28,28,1)
x_valid = x_valid.reshape(-1,28,28,1)
```

In [12]:
```
x_train
```

Out[12]:
```
array([[[[0.        ],
         [0.        ],
         [0.        ],
         ...,
         [0.        ],
         [0.        ],
         [0.        ]],

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

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

In [13]:
```
#building the CNN model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import (
    Dense,
    Conv2D,
    MaxPool2D,
    Flatten,
    Dropout,
    BatchNormalization,
)

model = Sequential()
model.add(Conv2D(255, (3, 3), strides=1, padding="same", activation="relu",
                 input_shape=(28, 28, 1)))
model.add(BatchNormalization())
model.add(ormalization())
model.add(MaxPool2D((2, 2), strides=2, padding="same"))
model.add(Conv2D(1MaxPool2D((2, 2), strides=2, padding="same"))
model.add(Conv2D(128, (3, 3), strides=1, padding="same", activation="relu"))
model.add(Dropout(0.3))
model.add(BatchN28, (3, 3), strides=1, padding="same", activation="relu"))
model.add(BatchNormalization())
#model.add(MaxPool2D((2, 2), strides=2, padding="same"))
model.add(Flatten())
model.add(Dense(units=255, activation="tanh"))
model.add(Dropout(0.3))
model.add(Dense(units=128, activation="tanh"))
model.add(Dense(units=num_classes, activation="softmax"))
```

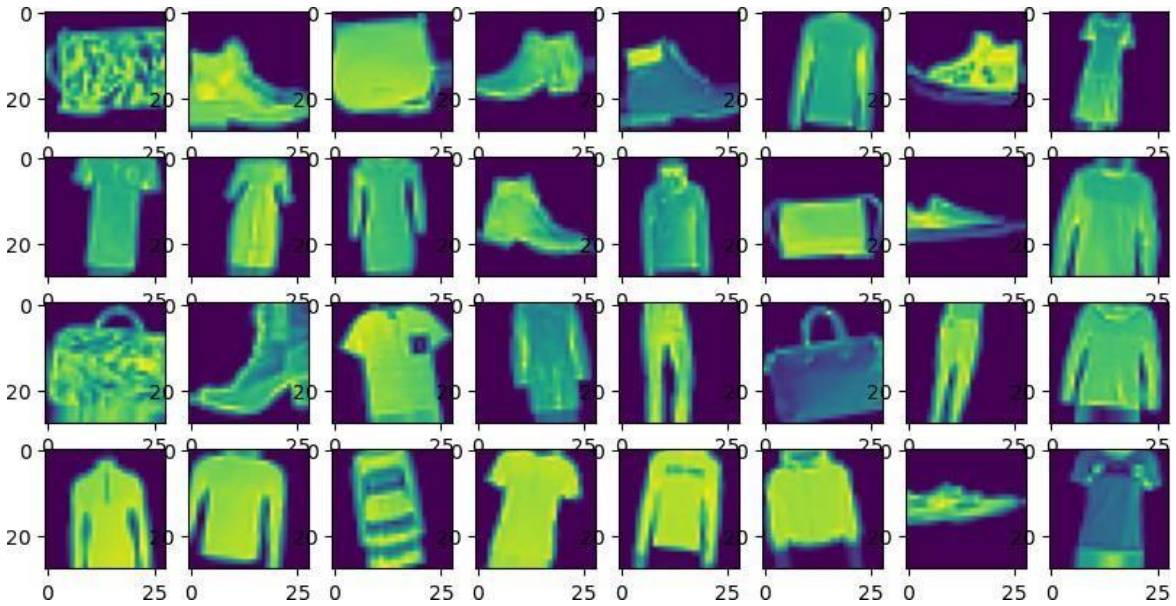
In [14]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 255)       2550

 batch_normalization (BatchN  (None, 28, 28, 255)      1020
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 14, 14, 255)      0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 128)       293888

 dropout (Dropout)           (None, 14, 14, 128)       0

 batch_normalization_1 (Batc  (None, 14, 14, 128)      512
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 128)         147584

 batch_normalization_2 (Batc  (None, 7, 7, 128)        512
 hNormalization)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 255)               1599615

 dropout_1 (Dropout)         (None, 255)               0

 dense_1 (Dense)             (None, 128)               32768

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 2,079,739
Trainable params: 2,078,717
Non-trainable params: 1,022
_________________________________________________________________
```

DATA Augmentation

In [15]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 1
    zoom_range=0.1,  # Randomly zoom image
    width_shift_range=0.1, # randomly shift images horizontally (fraction of
    height_shift_range=0.1, # randomly shift images vertically (fraction of t
    horizontal_flip=True, # randomly flip images horizontally
    vertical_flip=False, # Don't randomly flip images vertically
)
```

In [16]:
```
import matplotlib.pyplot as plt
import numpy as np
batch_size = 32
img_iter = datagen.flow(x_train, y_train, batch_size=batch_size)

x, y = img_iter.next()
fig, ax = plt.subplots(nrows=4, ncols=8,figsize = (10, 5))
for i in range(batch_size):
    image = x[i]
    ax.flatten()[i].imshow(np.squeeze(image))
plt.show()
```



In [17]:
```
datagen.fit(x_train)
```

Label---> description

1 : T-shirt/top
2 : Trouser
3 : Pullover
4 : Dress
5 : Coat
6 : Sandal
7 : Shirt
8 : Sneaker
9 : Bag
10: Ankle boot

In [18]:
```
model.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['accu
```

In [19]:
```
from tensorflow.keras import callbacks
early_stopping = callbacks.EarlyStopping(
    monitor='val_loss', min_delta=0.001, patience=5, restore_best_weights=True
```

In [20]:
```
history=model.fit(img_iter,
          epochs=20,
          steps_per_epoch=len(x_train)/batch_size,
          validation_data=(x_valid, y_valid),
          callbacks=[early_stopping])
```

```
Epoch 1/20

2023-03-14 21:19:47.109988: E tensorflow/core/grappler/optimizers/meta_optimi
zer.cc:954] layout failed: INVALID_ARGUMENT: Size of values 0 does not match
size of permutation 4 @ fanin shape insequential/dropout/dropout/SelectV2-2-T
ransposeNHWCToNCHW-LayoutOptimizer
```

```
1875/1875 [==============================] - 38s 16ms/step - loss: 0.6689 - a
ccuracy: 0.7436 - val_loss: 0.4479 - val_accuracy: 0.8286
Epoch 2/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.5046 - a
ccuracy: 0.8106 - val_loss: 0.3617 - val_accuracy: 0.8649
Epoch 3/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.4511 - a
ccuracy: 0.8318 - val_loss: 0.4987 - val_accuracy: 0.8111
Epoch 4/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.4250 - a
ccuracy: 0.8407 - val_loss: 0.3271 - val_accuracy: 0.8788
Epoch 5/20
1875/1875 [==============================] - 29s 16ms/step - loss: 0.4056 - a
ccuracy: 0.8481 - val_loss: 0.3768 - val_accuracy: 0.8533
Epoch 6/20
1875/1875 [==============================] - 29s 16ms/step - loss: 0.3906 - a
ccuracy: 0.8544 - val_loss: 0.2969 - val_accuracy: 0.8880
Epoch 7/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3854 - a
ccuracy: 0.8562 - val_loss: 0.3928 - val_accuracy: 0.8464
Epoch 8/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3766 - a
ccuracy: 0.8597 - val_loss: 0.3770 - val_accuracy: 0.8535
Epoch 9/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3731 - a
ccuracy: 0.8622 - val_loss: 0.3190 - val_accuracy: 0.8796
Epoch 10/20
1875/1875 [==============================] - 29s 16ms/step - loss: 0.3656 - a
ccuracy: 0.8648 - val_loss: 0.2765 - val_accuracy: 0.8991
Epoch 11/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3581 - a
ccuracy: 0.8678 - val_loss: 0.3157 - val_accuracy: 0.8791
Epoch 12/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3519 - a
ccuracy: 0.8709 - val_loss: 0.2671 - val_accuracy: 0.9022
Epoch 13/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3510 - a
ccuracy: 0.8697 - val_loss: 0.3222 - val_accuracy: 0.8745
Epoch 14/20
1875/1875 [==============================] - 28s 15ms/step - loss: 0.3498 - a
ccuracy: 0.8702 - val_loss: 0.3095 - val_accuracy: 0.8823
Epoch 15/20
1875/1875 [==============================] - 28s 15ms/step - loss: 0.3454 - a
ccuracy: 0.8725 - val_loss: 0.3076 - val_accuracy: 0.8827
Epoch 16/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3415 - a
ccuracy: 0.8740 - val_loss: 0.3512 - val_accuracy: 0.8705
Epoch 17/20
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3361 - a
ccuracy: 0.8758 - val_loss: 0.3579 - val_accuracy: 0.8629
```

In [21]:
```
print("ok")
```

```
ok
```

In [22]:
```
# retrain the CNN model for lower learning rate
opt = keras.optimizers.Adam(learning_rate=0.00001)
model.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['accu
early_stopping = callbacks.EarlyStopping(
    monitor='val_loss', patience=5, restore_best_weights=True)
history1=model.fit(img_iter,
          epochs=10,
          steps_per_epoch=len(x_train)/batch_size,
          validation_data=(x_valid, y_valid),
          callbacks=[early_stopping])
```

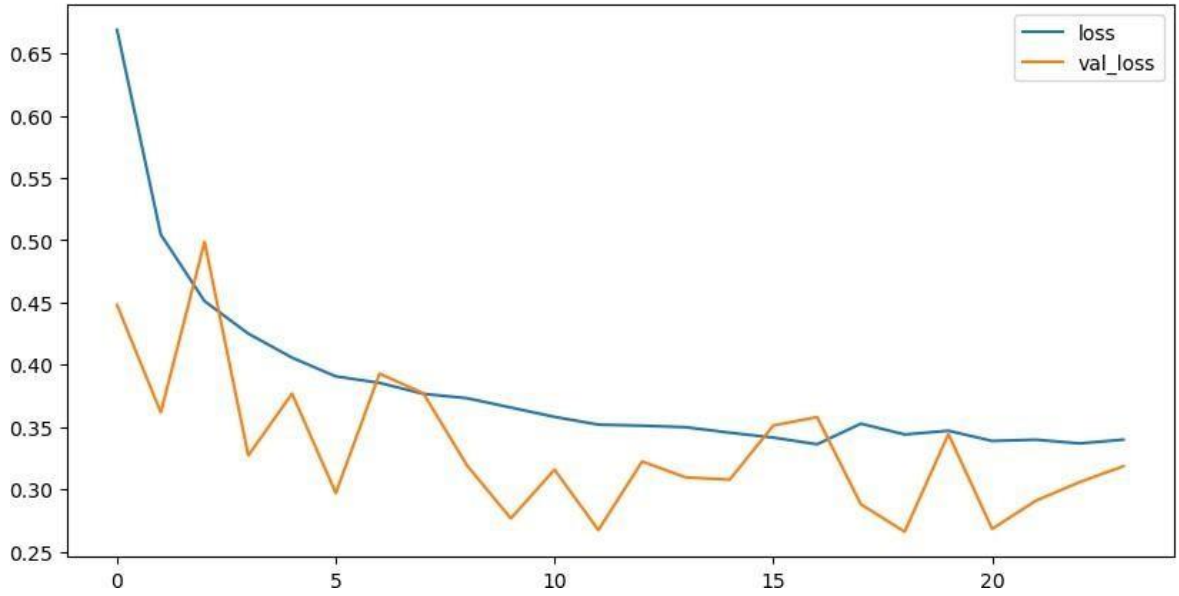
```
Epoch 1/10

2023-03-14 21:28:20.424052: E tensorflow/core/grappler/optimizers/meta_optimi
zer.cc:954] layout failed: INVALID_ARGUMENT: Size of values 0 does not match
size of permutation 4 @ fanin shape insequential/dropout/dropout/SelectV2-2-T
ransposeNHWCToNCHW-LayoutOptimizer

1875/1875 [==============================] - 33s 16ms/step - loss: 0.3526 - a
ccuracy: 0.8700 - val_loss: 0.2879 - val_accuracy: 0.8921
Epoch 2/10
1875/1875 [==============================] - 29s 15ms/step - loss: 0.3440 - a
ccuracy: 0.8730 - val_loss: 0.2658 - val_accuracy: 0.9016
Epoch 3/10
1875/1875 [==============================] - 29s 16ms/step - loss: 0.3469 - a
ccuracy: 0.8717 - val_loss: 0.3442 - val_accuracy: 0.8665
Epoch 4/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.3388 - a
ccuracy: 0.8742 - val_loss: 0.2681 - val_accuracy: 0.8977
Epoch 5/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.3398 - a
ccuracy: 0.8741 - val_loss: 0.2908 - val_accuracy: 0.8915
Epoch 6/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.3368 - a
ccuracy: 0.8761 - val_loss: 0.3057 - val_accuracy: 0.8864
Epoch 7/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.3398 - a
ccuracy: 0.8745 - val_loss: 0.3185 - val_accuracy: 0.8790
```

In [23]:
```
score = model.evaluate(x_valid, y_valid, verbose=0)
print('Test loss: %.5f' % score[0])
print('Test accuracy %.2f' % score[1])
```

```
Test loss: 0.26581
Test accuracy 0.90
```

In [24]:
```
first_training_history_df = pd.DataFrame(history.history)
second_training_history_df = pd.DataFrame(history1.history)
full_history_df = pd.concat([first_training_history_df, second_training_histor
                             , ignore_index=True,axis=0, sort=False)
full_history_df.loc[:, ['loss', 'val_loss']].plot(figsize = (10, 5))
print("Minimum Validation Loss: {:0.4f}".format(full_history_df['val_loss'].mi
```

```
Minimum Validation Loss: 0.2658
```



In [ ]:
```
```

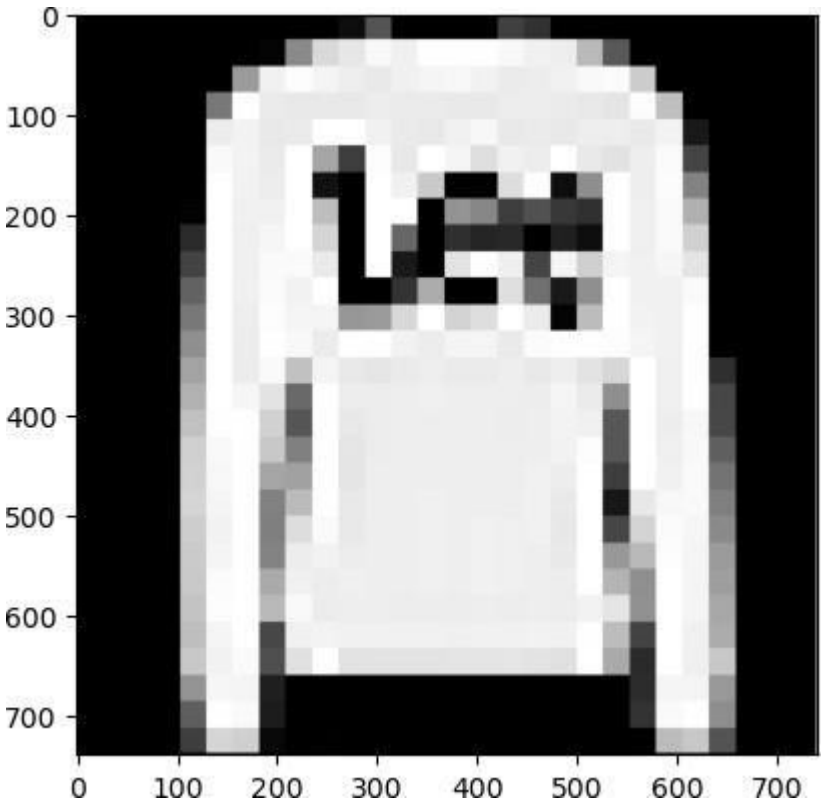
In [25]:
```
model.save('My_Fashion_MNist_model_v2')
```

In [26]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tensorflow.keras.preprocessing import image as image_utils
from tensorflow.keras.applications.imagenet_utils import preprocess_input

def show_image(image_path):
    image = mpimg.imread(image_path)
    plt.imshow(image)

def make_predictions(image_path):
    show_image(image_path)
    image = image_utils.load_img(image_path, grayscale=True, target_size=(28,
    image = image_utils.img_to_array(image)
    image = image.reshape(-1,28,28,1)
    image = image.astype('float32')
    image = image / 255.0
    preds = model.predict(image)
    return preds
```

In [27]:
```
labels = {
0:'T-shirt/top',
1:'Trouser',
2:'Pullover',
3:'ress',
4:'Coat',
5:'Sandal',
6:'Shirt',
7:'Sneaker',
8:'Bag',
9:'Ankle boot'
}
```

In [28]:
```
prdc=make_predictions('../input/fashion-clothing-classification/sample_image.p
```

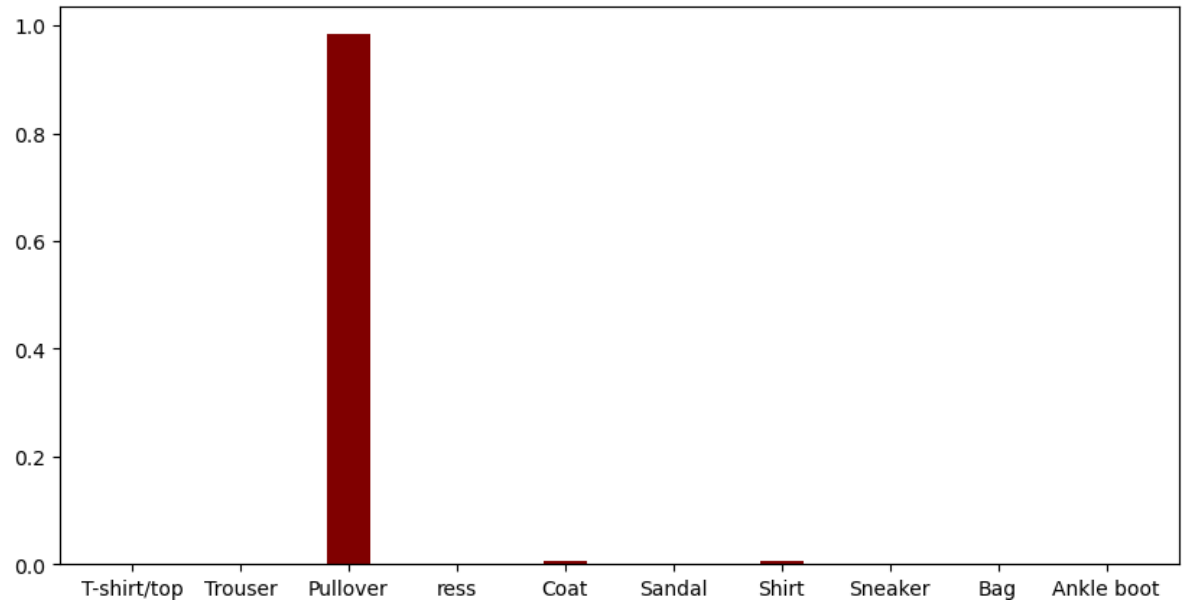
```
1/1 [==============================] - ETA: 0s

/opt/conda/lib/python3.7/site-packages/keras/utils/image_utils.py:410: UserWa
rning: grayscale is deprecated. Please use color_mode = "grayscale"
  'grayscale is deprecated. Please use color_mode = "grayscale"'

1/1 [==============================] - 0s 158ms/step
```



In [29]:
```
prdc
```

Out[29]:
```
array([[1.0175599e-03, 2.7550284e-05, 9.8515457e-01, 8.3116669e-04,
        6.4800354e-03, 4.5109547e-07, 5.9720599e-03, 3.1473706e-07,
        5.1605998e-04, 3.7849790e-07]], dtype=float32)
```

In [30]:
```
import matplotlib.pyplot as plt
%matplotlib inline

#plt.plot(list(labels.values()), list(prdc[0]))
fig = plt.figure(figsize = (10, 5))
plt.bar(list(labels.values()), list(prdc[0]), color ='maroon',
        width = 0.4)

plt.show()
```



In [31]:
```
target =labels[np.argmax(prdc[0])]
print("result label is ",target)
```

```
result label is  Pullover
```

In [ ]:
```

```

pretty nice our model did pridect the pullover properly