In [3]:
```
import os
import tensorflow as tf
from tensorflow import keras
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
np.random.seed(0)
import random
import tensorflow.keras as keras
from tensorflow.keras import datasets,layers,models
from tensorflow.keras.models import Sequential,load_model
from tensorflow.keras.layers import Dense,Dropout,Flatten
from tensorflow.keras.layers import Conv2D,MaxPool2D,BatchNormalization
from tensorflow.keras import backend as k
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import mnist
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau,ModelCheckpoint
from keras.models import model_from_json
from keras.utils.vis_utils import plot_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import validation_curve
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC
%matplotlib inline
```

In [5]:
```
(train_ing,train_labels),(test_ing,test_labels) = datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 1s 0us/step
```

In [6]:
```
train_ing,test_ing = train_ing/255.0, test_ing/255.0
```

In [7]:
```
len(train_ing)
```

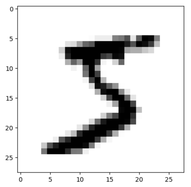
Out[7]:
```
60000
```

In [8]:
```
len(test_ing)
```

Out[8]:
```
10000
```

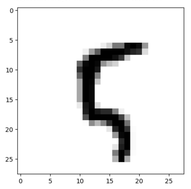
In [9]:
```
plt.imshow(train_ing[0],cmap=plt.cm.gray_r,interpolation='nearest')
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x240a2bc9590>
```



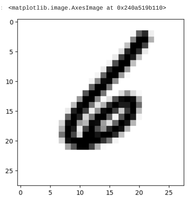
In [10]:
```
plt.imshow(train_ing[100],cmap=plt.cm.gray_r,interpolation='nearest')
```

Out[10]:
```
<matplotlib.image.AxesImage at 0x240a51540d0>
```



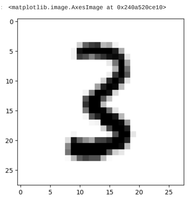
In [11]:
```
plt.imshow(train_ing[999],cmap=plt.cm.gray_r,interpolation='nearest')
```

Out[11]:
```
<matplotlib.image.AxesImage at 0x240a519c110>
```

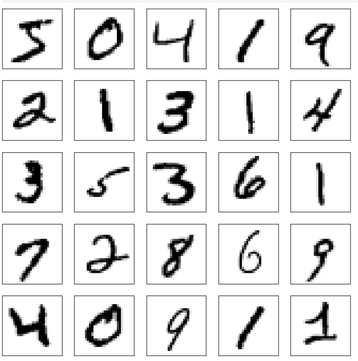


In [12]:
```
plt.imshow(train_ing[50000],cmap=plt.cm.gray_r,interpolation='nearest')
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x240a520ce10>
```



In [13]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_ing[i],cmap=plt.cm.binary)
plt.show()
```



In [14]:
```
model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28,28)),
        tf.keras.layers.Dense(128,activation='relu'),
        tf.keras.layers.Dense(10)
])
```

In [15]:
```
model.compile(
        optimizer=tf.keras.optimizers.Adam(0.001),
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
        metrics=['tf.keras.metrics.SparseCategoricalAccuracy()']
)
```

In [16]:
```
model.fit(
        train_ing,train_labels,
        epochs=10,
        validation_data=(test_ing,test_labels)
)
```

```
Epoch 1/10
1875/1875 [==============================] - 9s 2ms/step - loss: 0.2571 - sparse_categorical_accuracy: 0.9255 - val_loss: 0.1350 - val_sparse_categorical_accuracy: 0.9581
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1119 - sparse_categorical_accuracy: 0.9672 - val_loss: 0.1007 - val_sparse_categorical_accuracy: 0.9701
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0758 - sparse_categorical_accuracy: 0.9774 - val_loss: 0.0842 - val_sparse_categorical_accuracy: 0.9745
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0577 - sparse_categorical_accuracy: 0.9823 - val_loss: 0.0761 - val_sparse_categorical_accuracy: 0.9749
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0432 - sparse_categorical_accuracy: 0.9865 - val_loss: 0.0690 - val_sparse_categorical_accuracy: 0.9775
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0349 - sparse_categorical_accuracy: 0.9892 - val_loss: 0.0730 - val_sparse_categorical_accuracy: 0.9783
Epoch 7/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0273 - sparse_categorical_accuracy: 0.9920 - val_loss: 0.0707 - val_sparse_categorical_accuracy: 0.9775
Epoch 8/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0222 - sparse_categorical_accuracy: 0.9930 - val_loss: 0.0791 - val_sparse_categorical_accuracy: 0.9759
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0178 - sparse_categorical_accuracy: 0.9942 - val_loss: 0.0761 - val_sparse_categorical_accuracy: 0.9785
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0147 - sparse_categorical_accuracy: 0.9954 - val_loss: 0.0755 - val_sparse_categorical_accuracy: 0.9799
```

Out[16]:
```
<keras.callbacks.History at 0x240a571e990>
```

In [17]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

In [18]:
```
model.add(layers.Flatten())
```

In [19]:
```
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(10))
```

In [20]:
```
model.fit(train_ing,train_labels,epochs=12,validation_data=(test_ing,test_labels))
```

```
Epoch 1/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0136 - sparse_categorical_accuracy: 0.9954 - val_loss: 0.0805 - val_sparse_categorical_accuracy: 0.9797
Epoch 2/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0105 - sparse_categorical_accuracy: 0.9969 - val_loss: 0.0789 - val_sparse_categorical_accuracy: 0.9800
Epoch 3/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0092 - sparse_categorical_accuracy: 0.9971 - val_loss: 0.0870 - val_sparse_categorical_accuracy: 0.9788
Epoch 4/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0084 - sparse_categorical_accuracy: 0.9970 - val_loss: 0.0973 - val_sparse_categorical_accuracy: 0.9780
Epoch 5/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0085 - sparse_categorical_accuracy: 0.9972 - val_loss: 0.0947 - val_sparse_categorical_accuracy: 0.9787
Epoch 6/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0061 - sparse_categorical_accuracy: 0.9982 - val_loss: 0.0986 - val_sparse_categorical_accuracy: 0.9779
Epoch 7/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0081 - sparse_categorical_accuracy: 0.9973 - val_loss: 0.0995 - val_sparse_categorical_accuracy: 0.9792
Epoch 8/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0048 - sparse_categorical_accuracy: 0.9985 - val_loss: 0.0959 - val_sparse_categorical_accuracy: 0.9795
Epoch 9/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0051 - sparse_categorical_accuracy: 0.9984 - val_loss: 0.1022 - val_sparse_categorical_accuracy: 0.9799
Epoch 10/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0051 - sparse_categorical_accuracy: 0.9983 - val_loss: 0.1271 - val_sparse_categorical_accuracy: 0.9750
Epoch 11/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0056 - sparse_categorical_accuracy: 0.9983 - val_loss: 0.1177 - val_sparse_categorical_accuracy: 0.9760
Epoch 12/12
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0056 - sparse_categorical_accuracy: 0.9981 - val_loss: 0.1142 - val_sparse_categorical_accuracy: 0.9786
```

Out[20]:
```
<keras.callbacks.History at 0x240a5598650>
```

In [21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

 flatten_1 (Flatten)         (None, 10)                0

 dense_2 (Dense)             (None, 64)                704

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 103,124
Trainable params: 103,124
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [ ]: