Implementing Generative Adversarial Network on MNIST dataset

Import required libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

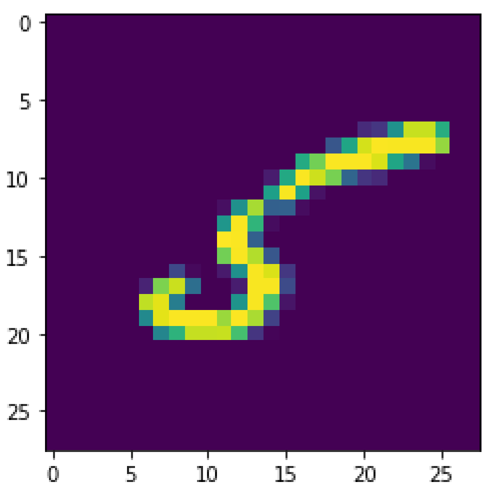
Load MNIST

In [2]:
```
from tensorflow.keras.datasets import mnist
```

In [3]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

In [4]:
```
plt.imshow(X_train[11])
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x1c68e906f88>
```



In [5]:
```
y_train[11]
```

Out[5]:
```
5
```

In [6]:
```
X_train[11].shape
```

Out[6]:
```
(28, 28)
```

Filter Data

In [7]:
```
y_train ==8
```

Out[7]:
```
array([False, False, False, ..., False, False,  True])
```

In [8]:
```
only_eight = X_train[y_train==8]
```

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

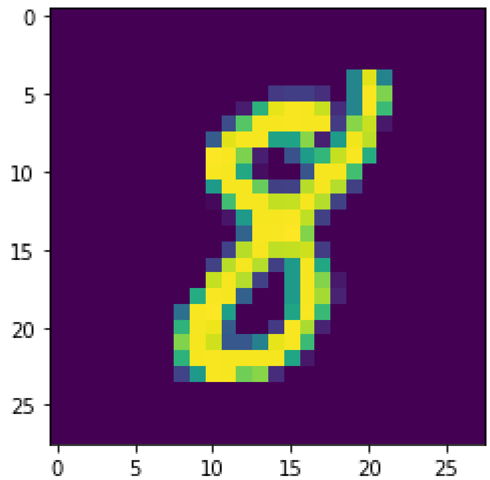
Out[9]: (5851, 28, 28)

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

Out[10]: (60000, 28, 28)

In [11]:
```
plt.imshow(only_eight[2])
```

Out[11]: <matplotlib.image.AxesImage at 0x1c68f997748>



Create Generator & Discriminator

In [12]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Reshape, Flatten
```

In [13]:
```
discriminator = Sequential()

discriminator.add(Flatten(input_shape=[28,28]))
discriminator.add(Dense(150, activation='relu'))
discriminator.add(Dense(100, activation='relu'))

#Final output layer
discriminator.add(Dense(1, activation='sigmoid'))

discriminator.compile(loss='binary_crossentropy', optimizer='adam')
```

In [14]:
```
#Choose a coding size, for 784 size we will choose around 100
coding_size = 100
```

In [15]:
```
generator = Sequential()
generator.add(Dense(100,activation='relu',input_shape=[coding_size]))
generator.add(Dense(150,activation='relu'))
generator.add(Dense(784,activation='relu'))

generator.add(Reshape([28,28]))
#We dont compile the generator compared to the discriminator
```

In [16]:
```
GAN = Sequential([generator,discriminator])
```

In [17]:
```
discriminator.trainable=False
```

```
#Discriminator should not be trained in the 2nd Phase
```

In [18]:
```
GAN.compile(loss='binary_crossentropy',optimizer='adam')
```

Create Training Batches

In [19]:
```
#Choose a smaller batch size for slow training
batch_size = 32
```

In [21]:
```
my_data = only_eight
type(my_data)
```

Out[21]:
```
numpy.ndarray
```

In [22]:
```
dataset = tf.data.Dataset.from_tensor_slices(my_data).shuffle(buffer_size=1000)
```

In [23]:
```
type(dataset)
```

Out[23]:
```
tensorflow.python.data.ops.dataset_ops.ShuffleDataset
```

In [24]:
```
dataset = dataset.batch(batch_size,drop_remainder=True).prefetch(1)
```

Check for details of GAN layers and model summary

In [25]:
```
epochs = 1
```

In [26]:
```
GAN
```

Out[26]:
```
<tensorflow.python.keras.engine.sequential.Sequential at 0x1c68ff8a608>
```

In [27]:
```
GAN.layers
```

Out[27]:
```
[<tensorflow.python.keras.engine.sequential.Sequential at 0x1c68ff8c548>,
 <tensorflow.python.keras.engine.sequential.Sequential at 0x1c68fe5b2c8>]
```

In [28]:
```
GAN.layers[0].layers
```

Out[28]:
```
[<tensorflow.python.keras.layers.core.Dense at 0x1c68fe3f948>,
 <tensorflow.python.keras.layers.core.Dense at 0x1c68ff7a548>,
 <tensorflow.python.keras.layers.core.Dense at 0x1c68ff9dc08>,
 <tensorflow.python.keras.layers.core.Reshape at 0x1c68ffc5888>]
```

In [29]:
```
GAN.layers[1].layers
```

Out[29]:
```
[<tensorflow.python.keras.layers.core.Flatten at 0x1c68fe5b548>,
 <tensorflow.python.keras.layers.core.Dense at 0x1c68fe47c48>,
 <tensorflow.python.keras.layers.core.Dense at 0x1c68fe9ef88>,
 <tensorflow.python.keras.layers.core.Dense at 0x1c68fee4f88>]
```

In [30]:
```
GAN.layers[0].summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 100)               10100
_________________________________________________________________
```

```
dense_4 (Dense)              (None, 150)               15150
_________________________________________________________________
dense_5 (Dense)              (None, 784)               118384
_________________________________________________________________
reshape (Reshape)            (None, 28, 28)            0
=================================================================
Total params: 143,634
Trainable params: 143,634
Non-trainable params: 0
_________________________________________________________________
```

In [31]:
```
GAN.layers[1].summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 150)               117750
_________________________________________________________________
dense_1 (Dense)              (None, 100)               15100
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
WARNING:tensorflow:Discrepancy between trainable weights and collected trainable weight
s, did you set `model.trainable` without calling `model.compile` after ?
Total params: 265,902
Trainable params: 132,951
Non-trainable params: 132,951
_________________________________________________________________
```

In [32]:
```
# Grab the seprate components
generator, discriminator = GAN.layers

# For every epcoh
for epoch in range(epochs):
    print(f"Currently on Epoch {epoch+1}")
    i = 0
    # For every batch in the dataset
    for X_batch in dataset:
        i=i+1
        if i%100 == 0:
            print(f"\tCurrently on batch number {i} of {len(my_data)//batch_size}")
        #####################################
        ## TRAINING THE DISCRIMINATOR ######
        ###################################

        # Create Noise
        noise = tf.random.normal(shape=[batch_size, coding_size])

        # Generate numbers based just on noise input
        gen_images = generator(noise)

        # Concatenate Generated Images against the Real Ones
        # TO use tf.concat, the data types must match!
        X_fake_vs_real = tf.concat([gen_images, tf.dtypes.cast(X_batch,tf.float32)], ax

        # Targets set to zero for fake images and 1 for real images
        y1 = tf.constant([[0.]] * batch_size + [[1.]] * batch_size)

        # This gets rid of a Keras warning
        discriminator.trainable = True
```

```
        # Train the discriminator on this batch
        discriminator.train_on_batch(X_fake_vs_real, y1)

        ## TRAINING THE GENERATOR     ######

        # Create some noise
        noise = tf.random.normal(shape=[batch_size, coding_size])

        # We want discriminator to belive that fake images are real
        y2 = tf.constant([[1.]] * batch_size)

        # Avois a warning
        discriminator.trainable = False

        GAN.train_on_batch(noise, y2)

print("TRAINING COMPLETE")
```

```
Currently on Epoch 1
        Currently on batch number 100 of 182
TRAINING COMPLETE
```

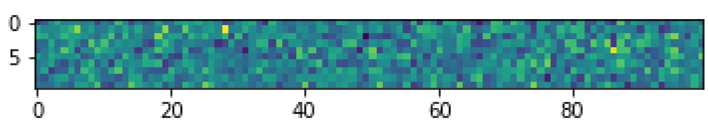
In [33]:
```
noise =  tf.random.normal(shape=[10,coding_size])
```

In [34]:
```
noise.shape
```

Out[34]:
```
TensorShape([10, 100])
```

In [35]:
```
plt.imshow(noise)
```

Out[35]:
```
<matplotlib.image.AxesImage at 0x1c691726088>
```



In [36]:
```
images = generator(noise)
images.shape
```

Out[36]:
```
TensorShape([10, 28, 28])
```

In [37]:
```
images[0]
```

Out[37]:
```
<tf.Tensor: id=72382, shape=(28, 28), dtype=float32, numpy=
array([[0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
       [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
       [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
```

```
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.8043506e+00,
         0.0000000e+00, 1.0909084e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 2.5751019e-01, 0.0000000e+00, 1.0447125e+00,
         1.1781101e+00, 0.0000000e+00, 1.7106526e+00, 0.0000000e+00,
         2.3266182e+00, 1.1143678e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 6.9041830e-04, 0.0000000e+00, 1.4659697e-01,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.9386269e-01,
         0.0000000e+00, 9.1958809e-01, 1.7012370e+00, 0.0000000e+00,
         1.3964919e+00, 0.0000000e+00, 1.5935283e+00, 0.0000000e+00,
         2.2601392e-02, 0.0000000e+00, 5.2309948e-01, 0.0000000e+00,
         0.0000000e+00, 8.3354533e-01, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         1.3800783e-01, 1.9158299e+00, 2.0188031e+00, 1.4451909e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 2.1988943e-02, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 1.3485229e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 1.1281419e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 1.0490597e+00, 0.0000000e+00,
         8.8424522e-01, 8.0346382e-01, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 3.0488652e-01, 5.2253526e-01, 0.0000000e+00,
         1.0244520e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 9.9003130e-01, 0.0000000e+00,
         0.0000000e+00, 9.0007287e-01, 8.8639289e-02, 0.0000000e+00,
         6.3908033e-02, 0.0000000e+00, 2.0485531e-01, 0.0000000e+00,
         0.0000000e+00, 5.2226090e-01, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 5.8084512e-01,
         2.6672062e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
```

```
         3.0366015e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         1.2234120e+00, 0.0000000e+00, 0.0000000e+00, 2.6790157e-01,
         3.2720545e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 3.2630417e-01,
         0.0000000e+00, 0.0000000e+00, 1.5484834e+00, 0.0000000e+00,
         1.4780517e+00, 3.9899278e-01, 0.0000000e+00, 2.0597284e+00,
         8.1485146e-01, 0.0000000e+00, 1.0693660e+00, 0.0000000e+00,
         1.2992555e+00, 3.2449418e-01, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         7.1940720e-01, 1.9360958e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 3.6758074e-01, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 3.5183394e-01, 1.0944357e+00, 0.0000000e+00,
         0.0000000e+00, 9.8866010e-01, 6.5411526e-01, 0.0000000e+00,
         9.1976309e-01, 0.0000000e+00, 1.2546607e+00, 0.0000000e+00,
         0.0000000e+00, 2.6911506e-02, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         1.1922400e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 5.9138656e-01,
         0.0000000e+00, 0.0000000e+00, 1.0324831e+00, 1.0710843e+00,
         5.4040563e-01, 6.4551234e-01, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         1.2271256e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 1.7747423e-01, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 1.4865626e-01, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         5.3351820e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 1.3439202e+00, 0.0000000e+00,
         8.3247948e-01, 7.0864141e-01, 0.0000000e+00, 0.0000000e+00,
         7.6118743e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 9.6911317e-01, 5.2443355e-02,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 2.7119491e-01, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.7844134e-01,
         6.1889511e-01, 0.0000000e+00, 0.0000000e+00, 1.4896040e-01,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         3.5216838e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.3444283e+00,
         0.0000000e+00, 0.0000000e+00, 9.4523251e-02, 0.0000000e+00,
         1.4768647e+00, 0.0000000e+00, 1.0739872e+00, 0.0000000e+00,
```

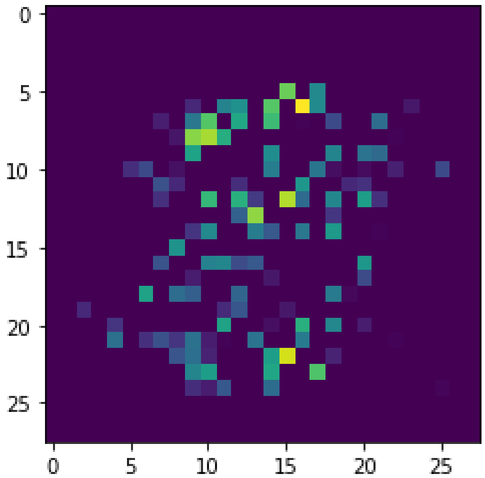
```
         1.8589982e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         8.6246938e-01, 0.0000000e+00, 3.2005316e-01, 5.7566667e-01,
         3.6007637e-01, 8.6518776e-01, 1.8371180e-01, 2.7199285e-02,
         0.0000000e+00, 8.8667649e-01, 0.0000000e+00, 0.0000000e+00,
         9.6537942e-01, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 1.8432740e-02, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         6.0975981e-01, 8.4913778e-01, 2.2255430e-01, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 1.3009639e+00, 2.1811633e+00,
         0.0000000e+00, 0.0000000e+00, 2.3180228e-01, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 1.0603122e+00, 1.3035382e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 1.3781748e+00, 0.0000000e+00,
         0.0000000e+00, 1.6848854e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 3.0001771e-01, 1.7671862e-01, 6.4178652e-01,
         0.0000000e+00, 0.0000000e+00, 8.2138282e-01, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 4.1902486e-02, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
        [0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
         0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00]],
       dtype=float32)>
```

In [38]:

```
plt.imshow(images[0])
```

Out[38]:

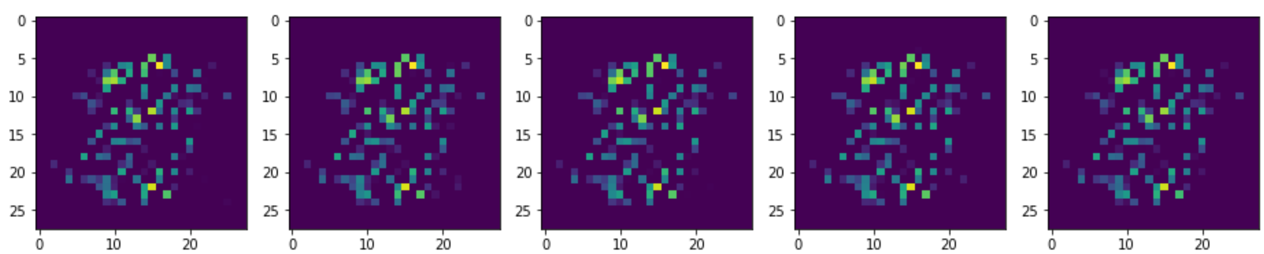
```
<matplotlib.image.AxesImage at 0x1c691368208>
```



In [44]:
```
plt.figure(figsize=(16,16))

plt.subplot(1,5,1)
plt.imshow(images[0])
plt.subplot(1,5,2)
plt.imshow(images[2])
plt.subplot(1,5,3)
plt.imshow(images[4])
plt.subplot(1,5,4)
plt.imshow(images[6])
plt.subplot(1,5,5)
plt.imshow(images[8])
```

Out[44]: <matplotlib.image.AxesImage at 0x1c6922eb488>



In [ ]: