Solução considerando 5 dimensões no Espaço Latente

In [1]:

```
import keras
from keras import layers
from keras import backend as K
from keras import optimizers
from keras.models import Model
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

img_shape = (28, 28, 1)
batch_size = 16
latent_dim = 5
```

```
/home/julio/Python/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In f
uture, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:

```
#signals used to train and test the model
name = "piminus_Pion-Eta0-PhiPiOver2-Energy50.npy"
data = np.load(name)
print(data.shape)
```

```
(20549, 28, 28)
```

In [3]:

```
limit = 20549
signal = np.zeros(limit)
for i in range(limit):
    signal[i] = np.sum(data[i].flatten())
```

In [4]:

```
temp = 0
for i in range(limit):
  #  print("Enegy of signal " + str(i) + ": " + str(signal[i]))
    temp  = temp + signal[i]
print("Energia do Sinal Médio >>>>>>>>>>>>> " + str(temp/20549))
```

```
Energia do Sinal Médio >>>>>>>>>>>>> 11.139551370775349
```

In [5]:

```
print("Sinal usado neste ensaio : " + name)
```

```
Sinal usado neste ensaio : piminus_Pion-Eta0-PhiPiOver2-Energy50.npy
```

In [6]:

```
print("Maxima Energia: "+str(np.max(signal)))
```

```
Maxima Energia: 49.12188720703125
```

In [7]:

```
print("Mínima Energia: "+str(np.min(signal)))
```

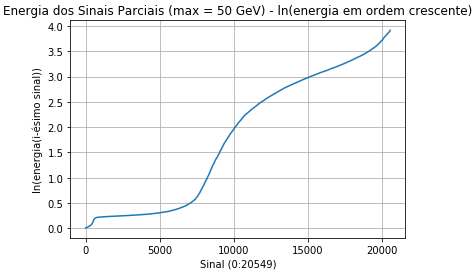
```
Mínima Energia: 0.010009774006903172
```

In [8]:

```
plt.title("Energia dos Sinais Parciais (max = 50 GeV) - energia em ordem crescente")
plt.xlabel("Sinal (0:20549)")
plt.ylabel("energia(i-ésimo sinal)")
plt.grid()
plt.plot((np.sort(signal)))
plt.show()
```

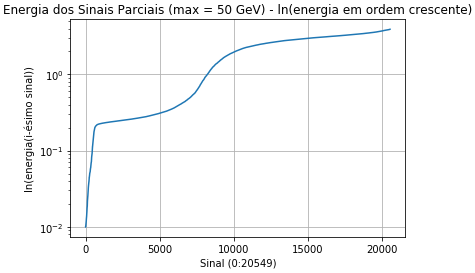
In [9]:

```
plt.title("Energia dos Sinais Parciais (max = 50 GeV) - ln(energia em ordem crescente)")
plt.xlabel("Sinal (0:20549)")
plt.ylabel("ln(energia(i-ésimo sinal))")
plt.grid()
plt.plot(np.log1p(np.sort(signal)))
plt.show()
```



In [10]:

```
plt.title("Energia dos Sinais Parciais (max = 50 GeV) - ln(energia em ordem crescente)")
plt.xlabel("Sinal (0:20549)")
plt.ylabel("ln(energia(i-ésimo sinal))")
plt.grid()
plt.semilogy()
plt.plot(np.log1p(np.sort(signal)))
plt.show()
```



In [11]:

```
#Redução do Conjunto de Dados (Apenas os sinais com Enegia <= 0.5 Gev)
```

In [12]:

```
data_t = data[signal < 0.5]
```

In [13]:

```
data_t.shape #6399/20549  Sinais parciais que serão usados na simulacao
```

Out[13]:

```
(6399, 28, 28)
```

In [14]:

```
limit = data_t.shape[0]
print(limit)
signal_t = np.zeros(limit) #6399 componentes
for i in range(limit): #0 a 6398
    signal_t[i] = np.sum(data_t[i].flatten()) #cada componente do sinal (i) acumula a energia em data_t[i] apenas sinais com menos de 0.5GeV
```

```
6399
```

In [15]:

```
print("Maxima Energia: "+str(np.max(signal_t))) #confirma que estamos trabalhando na faixa Gev < 0.5
```

```
Maxima Energia: 0.499929815530777
```

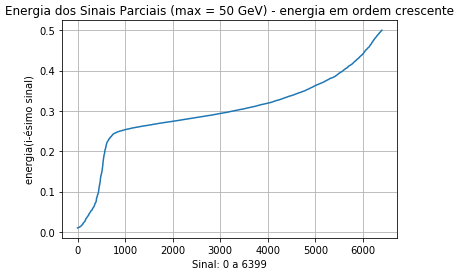
In [16]:

```
print("Mínima Energia: "+str(np.min(signal_t))) # Gev < 0.5 (descartamos todos os demais gerais de nossa massa de testes)
```

```
Mínima Energia: 0.010009774006903172
```

In [17]:

```
plt.title("Energia dos Sinais Parciais (max = 50 GeV) - energia em ordem crescente")
plt.xlabel("Sinal: 0 a 6399")
plt.ylabel("energia(i-ésimo sinal)")
plt.grid()
plt.plot((np.sort(signal_t)))
plt.show()
```



In [18]:

```
#Simulação com o sub_conjunto de dados
```

In [19]:

```
input_img = keras.Input(shape=img_shape)
x = layers.Conv2D(80, 3,padding='same', activation='relu')(input_img)
x = layers.Conv2D(32, 3,padding='same', activation='relu',strides=(2, 2))(x)
x = layers.Conv2D(32, 3,padding='same', activation='relu')(x)
x = layers.Conv2D(32, 3,padding='same', activation='relu')(x)
shape_before_flattening = K.int_shape(x)
x = layers.Flatten()(x)
x = layers.Dense(80, activation='relu')(x) # 32 para 80
z_mean = layers.Dense(latent_dim)(x) # 80 / 5  = 16 batch
z_log_var = layers.Dense(latent_dim)(x)
```

In [20]:

```
def sampling(args):
    z_mean, z_log_var = args
    epsilon = K.random_normal(shape=(K.shape(z_mean)[0], latent_dim), mean=0., stddev=1.)
    return z_mean + K.exp(z_log_var) * epsilon
z = layers.Lambda(sampling)([z_mean, z_log_var])
```

In [21]:

```
decoder_input = layers.Input(K.int_shape(z)[1:])
x = layers.Dense(np.prod(shape_before_flattening[1:]),activation='relu')(decoder_input)
x = layers.Reshape(shape_before_flattening[1:])(x)
x = layers.Conv2DTranspose(80, 3,padding='same',activation='relu',strides=(2, 2))(x) # 32 para 80
x = layers.Conv2D(1, 3,padding='same',activation='sigmoid')(x)
decoder = Model(decoder_input, x)
z_decoded = decoder(z)
```

In [22]:

```
class CustomVariationalLayer(keras.layers.Layer):

    def vae_loss(self, x, z_decoded):
        x = K.flatten(x)
        z_decoded = K.flatten(z_decoded)
        xent_loss = keras.metrics.binary_crossentropy(x, z_decoded)
        kl_loss = -5e-4 * K.mean(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
        return K.mean(xent_loss + kl_loss)

    def call(self, inputs):
        x = inputs[0]
        z_decoded = inputs[1]
        loss = self.vae_loss(x, z_decoded)
        self.add_loss(loss, inputs=inputs)
        return x
y = CustomVariationalLayer()([input_img, z_decoded])
```

In [23]:

```
vae = Model(input_img, y)
vae.compile(optimizer='rmsprop', loss=None)
```

In [24]:

```
#Massa de Dados para Simulação: data_t, ou seja sinais com Energia <= 0.5 Gev
data_t.shape
print("20% de validacao")
data_t_train = data_t[:5119,:,:]
data_t_test = data_t[5119:,:,:]
```

```
20% de validacao
```

In [25]:

```
data_t.shape
```

Out[25]:

```
(6399, 28, 28)
```

In [26]:

```
data_t_train.shape
```

Out[26]:

```
(5119, 28, 28)
```

In [27]:

```
data_t_test.shape
```

Out[27]:

```
(1280, 28, 28)
```

In [28]:

```
data_t_train = data_t_train.reshape(data_t_train.shape + (1,))
print(data_t_train.shape)
data_t_test = data_t_test.reshape(data_t_test.shape + (1,))
print(data_t_test.shape)
vae.fit(x=data_t_train, y=None,shuffle=True,epochs=3,batch_size=100,validation_data=(data_t_test, None))
```

```
(5119, 28, 28, 1)
(1280, 28, 28, 1)
Train on 5119 samples, validate on 1280 samples
Epoch 1/3
5119/5119 [==============================] - 16s 3ms/step - loss: 0.0806 - val_loss: 0.0076
Epoch 2/3
5119/5119 [==============================] - 15s 3ms/step - loss: 0.0035 - val_loss: 0.0019
Epoch 3/3
5119/5119 [==============================] - 15s 3ms/step - loss: 0.0017 - val_loss: 0.0016
```

Out[28]:

```
<keras.callbacks.History at 0x7f6385bad518>
```

In [29]:

```
np.sort(signal_t).shape #Sinais Originais com Energia < 0.5
```

Out[29]:

```
(6399,)
```

In [30]:

```
sreal_t = np.zeros(784) #Sinal real gerado por todos os sinais com energia < 0.5 Gev
limit = data_t.shape[0]
print(limit) #limit = 6399
for i in range(limit):
    sreal_t = sreal_t + data_t[i].flatten() #integração dos sinais parciais 0 a 6398, Energia < 0.5 GeV
```

```
6399
```

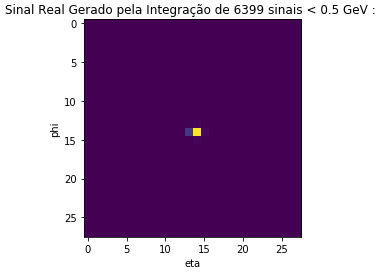
In [31]:

```
#Sinal Real com Energia Inferior a 0.5 Gev
print("Energia do Sinal Real Gerado pela Integração de 6399 sinais < 0.5 GeV : "+str(np.sum(sreal_t)/6399))
```

```
Energia do Sinal Real Gerado pela Integração de 6399 sinais < 0.5 GeV : 0.2997009527403352
```

In [32]:

```
plt.title("Sinal Real Gerado pela Integração de 6399 sinais < 0.5 GeV : ")
plt.xlabel("eta")
plt.ylabel("phi")
plt.imshow(sreal_t.reshape(28,28))
plt.show()
```



In [33]:

```
#Contrução de um Sinal Sintético usando os sinais gerado a partir do espaço latente do autoenconder.
limit_i = 5
limit_j = 5
limit_k = 5
limit_l = 5
limit_m = 5
signal_counter = 0
synthetic_signal = np.zeros((1,28,28))
partial_signals = np.zeros((limit_i*limit_j*limit_k*limit_l*limit_m,28,28))
print(synthetic_signal.shape)
print(partial_signals.shape)

for i in range(limit_i):
    for j in range(limit_j):
        for k in range(limit_k):
            for l in range(limit_l):
                for m in range(limit_m):
                    z_sample = np.array([[i,j,k,l,m]])
                    z_sample = np.tile(z_sample, batch_size).reshape(batch_size, 5) # 2 para 5
                    x_decoded = decoder.predict(z_sample, batch_size=batch_size)
                    partial_signals[signal_counter,:,:] = x_decoded[0].reshape(28,28)
                    signal_counter = signal_counter + 1
                   # print(signal_counter)
                    synthetic_signal = synthetic_signal + x_decoded[0].reshape(28,28)
#print(synthetic_signal.shape)
```

```
(1, 28, 28)
(3125, 28, 28)
```

In [34]:

```
print("Numero de Sinais Sintéticos Parciais Gerados a partir do Espaço Latente : "+str(partial_signals.shape))
```

```
Numero de Sinais Sintéticos Parciais Gerados a partir do Espaço Latente : (3125, 28, 28)
```

In [35]:

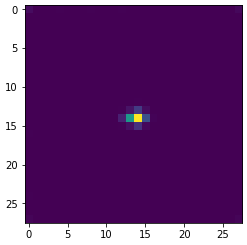
```
print("Sinal Sintetico Contruido pela Integração dos 6400 Sinais Sintéticos Parciais: "+(str(synthetic_signal.shape)))
```

```
Sinal Sintetico Contruido pela Integração dos 6400 Sinais Sintéticos Parciais: (1, 28, 28)
```

In [36]:

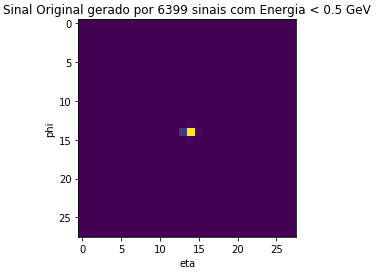
```
print("Sinal Sintético (ln)formado por:"+str(limit_i*limit_j*limit_k*limit_l*limit_m)+" amostras")
plt.imshow(np.log1p(synthetic_signal[0,:,:]))
plt.show()
```

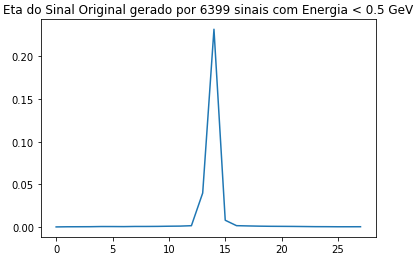
```
Sinal Sintético (ln)formado por:3125 amostras
```

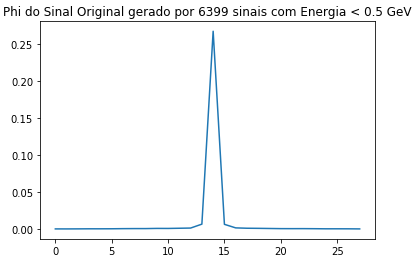


In [37]:

```
#Eta e Phi do Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV
real = sreal_t.reshape(28,28)
plt.title("Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV")
plt.xlabel("eta")
plt.ylabel("phi")
plt.imshow(real)
plt.show()

plt.title("Eta do Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV")
eta = np.sum(real/6400,axis=0)
plt.plot(eta)
plt.show()

plt.title("Phi do Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV")
phi = np.sum(real/6400,axis=1)
plt.plot(phi)
plt.show()
```

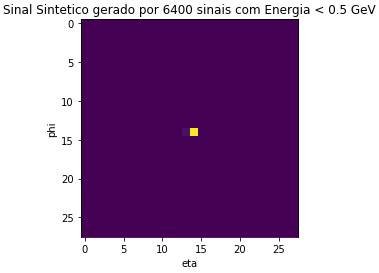


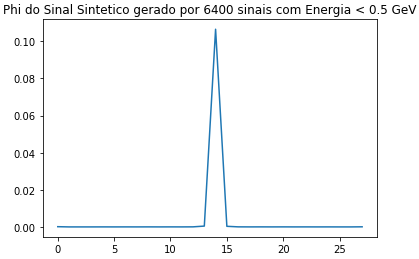




In [38]:

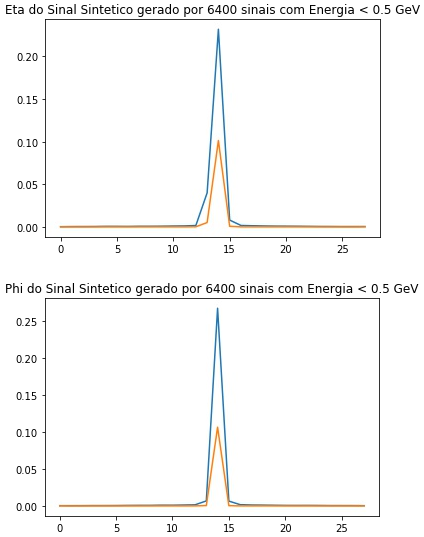
```
syn = synthetic_signal.reshape(28,28)
plt.title("Sinal Sintetico gerado por 6400 sinais com Energia < 0.5 GeV")
plt.xlabel("eta")
plt.ylabel("phi")
plt.imshow(syn)
plt.show()

plt.title("Eta do Sinal Sintetico gerado por 6400 sinais com Energia < 0.5 GeV")
eta = np.sum(syn/6400,axis=0)
plt.plot(eta)
plt.show()

plt.title("Phi do Sinal Sintetico gerado por 6400 sinais com Energia < 0.5 GeV")
phi = np.sum(syn/6400,axis=1)
plt.plot(phi)
plt.show()
```



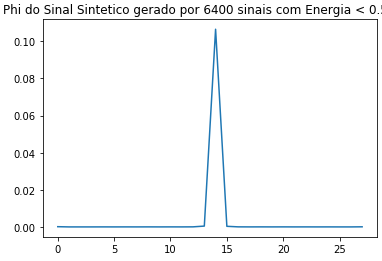


In [39]:

```
#Sobreposicao
plt.title("Eta do Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV")
eta = np.sum(real/6400,axis=0)
plt.plot(eta)

plt.title("Eta do Sinal Sintetico gerado por 6400 sinais com Energia < 0.5 GeV")
eta = np.sum(syn/6400,axis=0)
plt.plot(eta)

plt.show()


plt.title("Phi do Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV")
phi = np.sum(real/6400,axis=1)
plt.plot(phi)

plt.title("Phi do Sinal Sintetico gerado por 6400 sinais com Energia < 0.5 GeV")
phi = np.sum(syn/6400,axis=1)
plt.plot(phi)

plt.show()
```

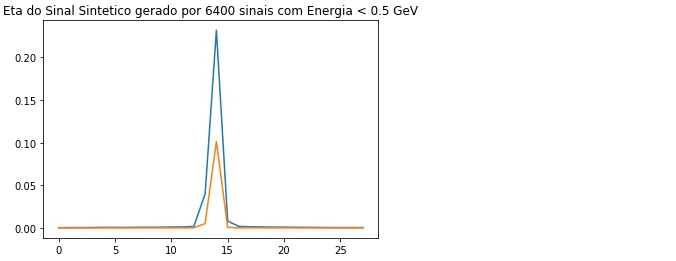


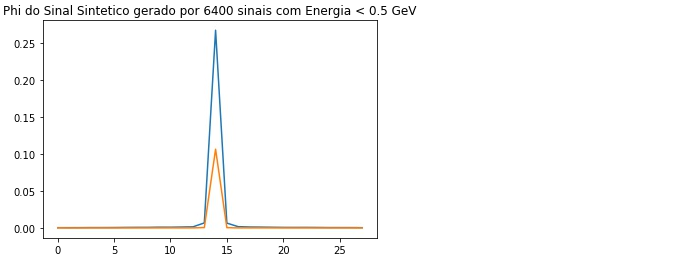
<] .btn-link:focus { color: #777777; text-decoration: none; } .btn-lg, .btn-group-lg/div>
TgRLwwhqSfi3wjjLD3x7HVebuVf+ABwk34ezww4BBYAToB56Kw18H/BdQBLYC66dY/oclF/HDwLsABw6J4xYBXwZ2AL8DLgLaKiznl8AbqqwnvdxrgCuAfwb6gNuB56am/WxMexG4A3hZatx64KsV1vE54DNV0nArcE78fhbwY+BvgJ3Ab4FT4FT4rjL4r
" >



In [39]:

```
#Sobreposicao
plt.title("Eta do Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV")
eta = np.sum(real/6400,axis=0)
plt.plot(eta)

plt.title("Eta do Sinal Sintetico gerado por 6400 sinais com Energia < 0.5 GeV")
eta = np.sum(syn/6400,axis=0)
plt.plot(eta)

plt.show()


plt.title("Phi do Sinal Original gerado por 6399 sinais com Energia < 0.5 GeV")
phi = np.sum(real/6400,axis=1)
plt.plot(phi)

plt.title("Phi do Sinal Sintetico gerado por 6400 sinais com Energia < 0.5 GeV")
phi = np.sum(syn/6400,axis=1)
plt.plot(phi)

plt.show()
```





<] .btn-link:focus { color: #777777; text-decoration: none; } .btn-lg, .btn-group-lg