

# DENOISING AUTOENCODERS

Se quitó el ruido de imágenes de texto usando tres arquitecturas diferentes:

1. MLP
2. CNN + post process
3. CNN + 2 post process

# Datos

Los datos de entrenamiento están conformados con fotos del mismo texto, con diferentes tipos de daño y fuente.

There are several classic spatial filters  
high frequency noise from images. The mean  
and the closing opening filter are frequently  
is a lowpass or smoothing filter that replace  
with the neighborhood mean. It reduces the  
the image edges. The median filter calculate  
pixel neighborhood for each pixel, thereby re  
effect. Finally, the opening closing filter  
filter that combines the same number of eros  
operations in order to eliminate small object

The main goal was to train a neural network  
to obtain a clean image from a noisy one. In  
it was much easier to obtain a simulated noisy  
one than to clean a subset of noisy images.  
simulated noisy images follows this scheme:

There are several classic spatial filters  
high frequency noise from images. The mean  
and the closing opening filter are frequently  
is a lowpass or smoothing filter that replace  
with the neighborhood mean. It reduces the  
the image edges. The median filter calculate  
pixel neighborhood for each pixel, thereby re  
effect. Finally, the opening closing filter  
filter that combines the same number of eros  
operations in order to eliminate small object

The main goal was to train a neural network  
to obtain a clean image from a noisy one. In  
it was much easier to obtain a simulated noisy  
one than to clean a subset of noisy images.  
simulated noisy images follows this scheme:

There exist several methods to design  
fields to be filled in. For instance, fields may be  
surrounded by bounding boxes. These may be  
by guiding rulers. These may be  
write and, therefore, minimize  
d overlapping with other parts  
ides can be located on a separate  
at is located below the form  
rectly on the form. The use of  
ed is much better than the

 122.png

There are several classic spatial filters  
high frequency noise from images. The mean  
and the closing opening filter are frequently  
is a lowpass or smoothing filter that replace  
with the neighborhood mean. It reduces the  
the image edges. The median filter calculate  
pixel neighborhood for each pixel, thereby re  
effect. Finally, the opening closing filter  
filter that combines the same number of eros  
operations in order to eliminate small object

 120.png

There exist several methods to design  
filled in. For instance, fields may be  
boxes, by light rectangles or by guiding  
specify where to write and, therefore  
skew and overlapping with other parts  
ides can be located on a separate  
ated below the form or they can be  
m. The use of guides on a separate  
m the point of view of the quality  
requires giving more instructions

 119.png

There exist several methods to design  
filled in. For instance, fields may be  
boxes, by light rectangles or by guiding  
specify where to write and, therefore  
skew and overlapping with other parts  
ides can be located on a separate  
ated below the form or they can be  
m. The use of guides on a separate  
m the point of view of the quality  
requires giving more instructions

 113.png

There are several classic spatial filters  
high frequency noise from images. The mean  
and the closing opening filter are frequently  
is a lowpass or smoothing filter that replace  
with the neighborhood mean. It reduces the  
the image edges. The median filter calculate  
pixel neighborhood for each pixel, thereby re  
effect. Finally, the opening closing filter  
filter that combines the same number of eros  
operations in order to eliminate small object

 111.png

There exist several methods to design  
filled in. For instance, fields may be  
boxes, by light rectangles or by guiding  
specify where to write and, therefore  
skew and overlapping with other parts  
ides can be located on a separate  
ated below the form or they can be  
m. The use of guides on a separate  
m the point of view of the quality  
requires giving more instructions

 110.png

# Arquitectura 1: MLP

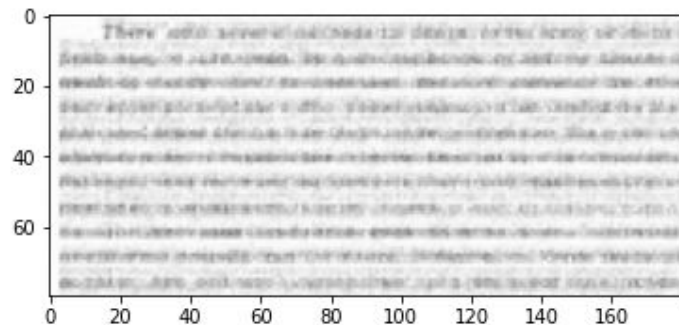
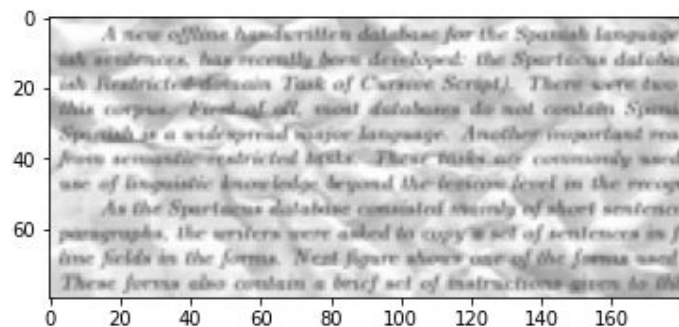
- No muy efectiva
- Overfitting

```
37 class Autocoder(nn.Module):
38     def __init__(self):
39         super(Autocoder, self).__init__()
40         self.encoder = Encoder()
41         self.Decoder = Decoder()
42
43     def forward(self, image):
44         z = self.encoder(image)
45         out = self.Decoder(z)
46         return out
47
48     def reiniciar(self):
49         super(Autocoder, self).__init__()
50         self.encoder = Encoder()
51         self.Decoder = Decoder()
```

```
1 class Encoder(nn.Module):
2     def __init__(self):
3         super(Encoder, self).__init__()
4         self.f1 = nn.Sequential(
5             nn.Linear(180*80, 180*10),
6             nn.ReLU()
7         )
8
9         self.f2 = nn.Sequential(
10             nn.Linear(180*10, 180*10),
11             nn.ReLU()
12         )
13
14     def forward(self, image):
15         out = self.f1(image)
16         z = self.f2(out)
17         return z
18
19 class Decoder(nn.Module):
20     def __init__(self):
21         super(Decoder, self).__init__()
22
23         self.f1 = nn.Sequential(
24             nn.Linear(180*10, 180*10),
25             nn.ReLU()
26         )
27
28         self.f5 = nn.Sequential(
29             nn.Linear(180*10, 180*80),
30         )
31
32     def forward(self, z):
33         out = self.f1(z)
34         out = torch.tanh(self.f5(z))
35         return out
```

## Resultados

Se puede notar que hay un overfitting, ya que después de pasar por el autoencoder, se nota en la palabra inicial “There”, el cual es parte del texto de entrenamiento. La imagen que se esperaba debía empezar con “A new”.



## Arquitectura 2: CNN

La arquitectura 2 es una CNN autoencoder que logra eliminar el ruido, pero produce un texto no estético, por ello se decide aumentar algunas capas de neuronas posteriores al decoder denominadas Process, cuya función es que la imagen saliente sea estética.

```
class Autoncoder(nn.Module):
    def __init__(self):
        super(Autoncoder,self).__init__()
        self.encoder = Encoder()
        self.Decoder = Decoder()
        self.Process = Process()
    def forward(self, image):
        z = self.encoder(image)
        out = self.Decoder(z)
        out = self.Process(out)
        return out
    def reiniciar(self):
        super(Autoncoder,self).__init__()
        self.encoder = Encoder()
        self.Decoder = Decoder()
        self.Process = Process()
```

# Resultados

Imagen original:

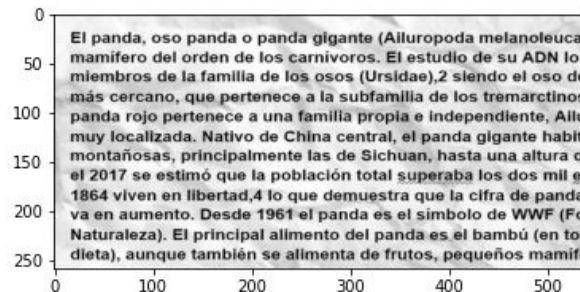
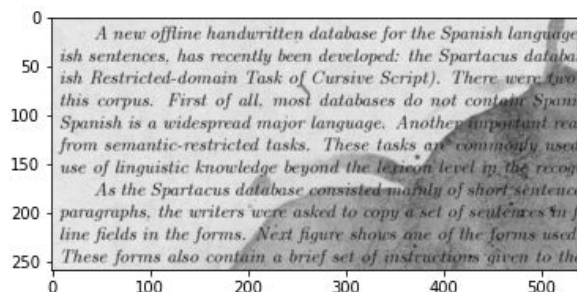


Imagen después  
del autoencoder:

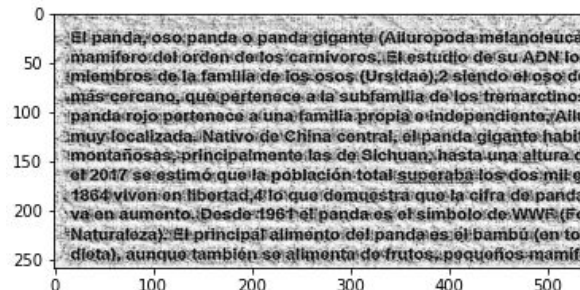
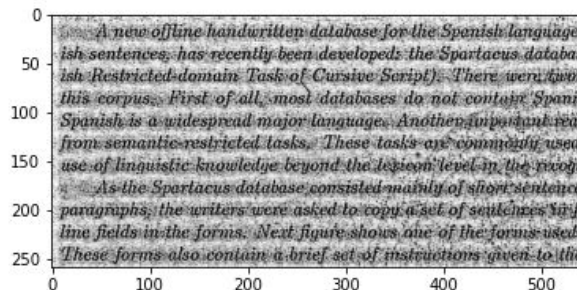
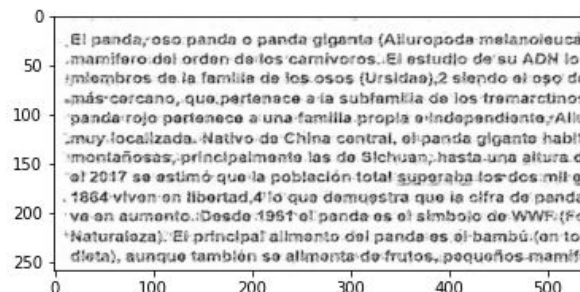
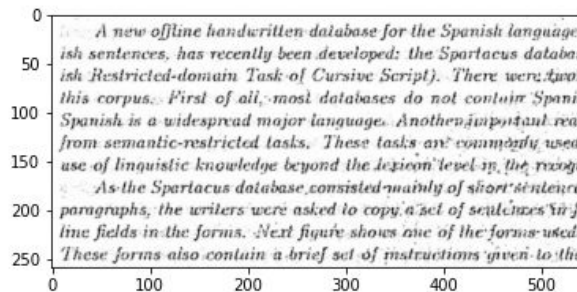


Imagen después de  
Process:



Como se puede observar, el autoencoder hace que el fondo de la imagen sea homogénea y resalta el texto, y en la última imagen las líneas de dibujo.

Luego de esto, Process limpia el fondo.

Imagen original:

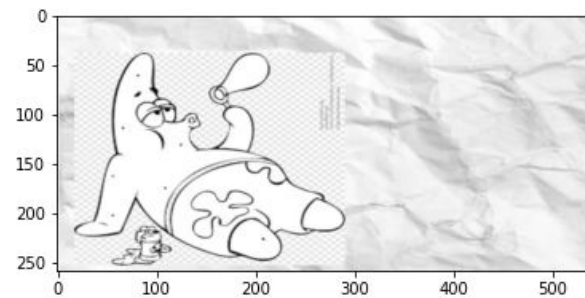


Imagen después del autoencoder:

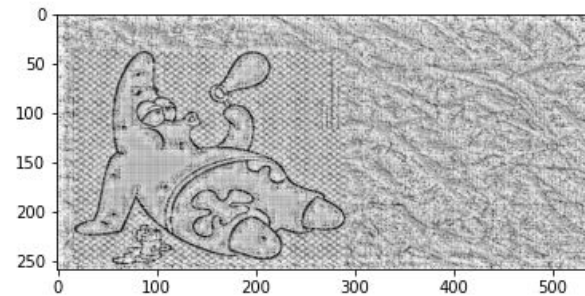
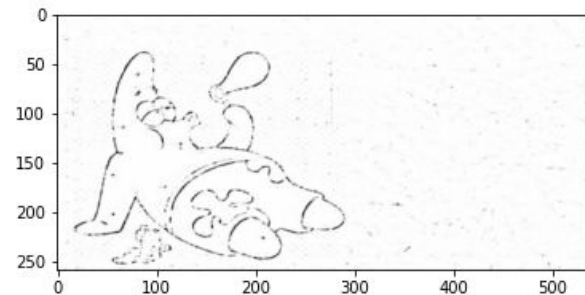


Imagen después de Process:





## Arquitectura 3: CNN

Esta arquitectura sigue la misma idea de la anterior pero duplica Process para mejorar los resultados, lo cual logra con éxito.

Se debe tener en cuenta que ambos Process son entrenados en conjunto para poder lograr limpiar el fondo de las imágenes. Por ello, el resultado de cada uno será diferente al Process original

```
class Autoncoder(nn.Module):  
    def __init__(self):  
        super(Autoncoder, self).__init__()  
        self.encoder = Encoder()  
        self.Decoder = Decoder()  
        self.Process = Process()  
        self.Process2 = Process()  
    def forward(self, image):  
        z = self.encoder(image)  
        out = self.Decoder(z)  
        out = self.Process(out)  
        out = self.Process2(out)  
        return out  
    def reiniciar(self):  
        super(Autoncoder, self).__init__()  
        self.encoder = Encoder()  
        self.Decoder = Decoder()  
        self.Process = Process()  
        self.Process2 = Process()
```



# Resultados

Imagen original:

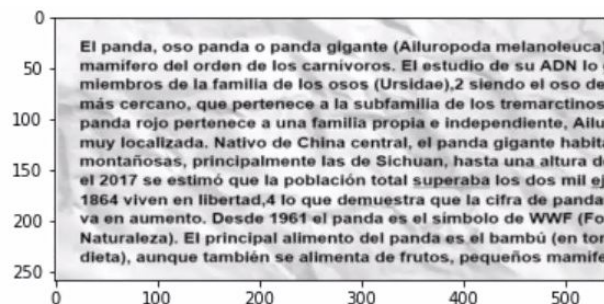


Imagen después del autoencoder:

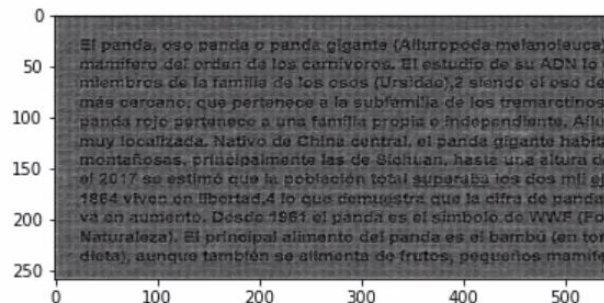


Imagen después del primer Process:

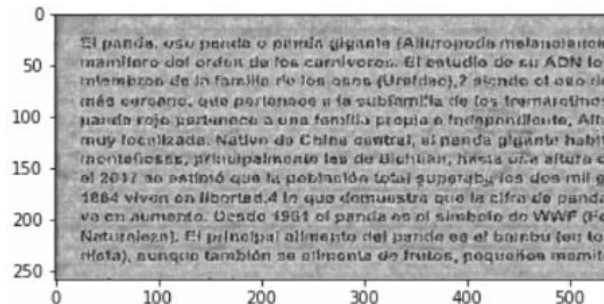


Imagen después del segundo process:

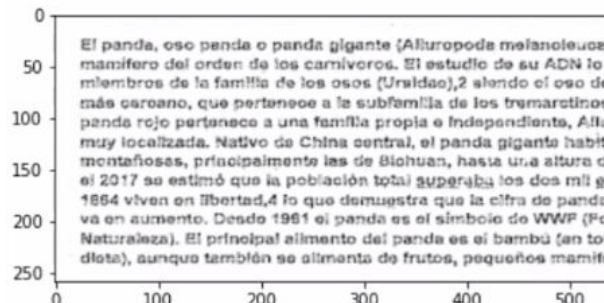


Imagen original:

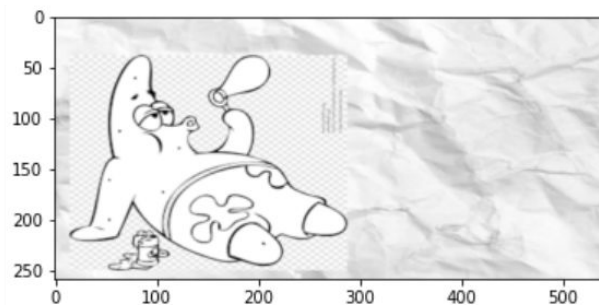


Imagen después  
del autoencoder:

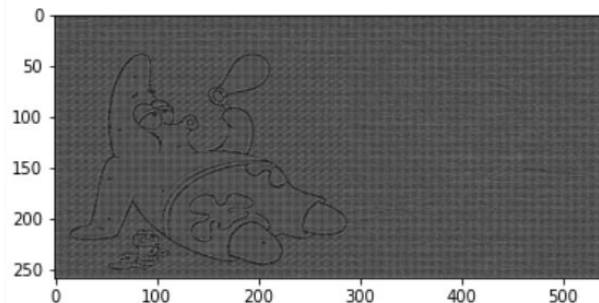


Imagen después  
del primer  
Process:

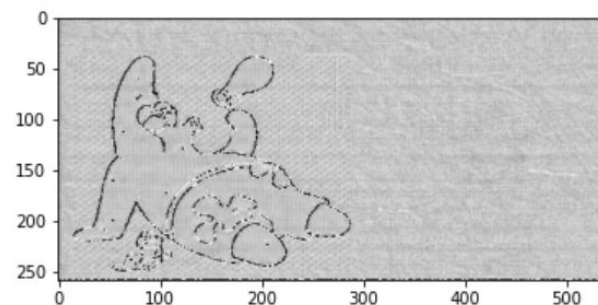
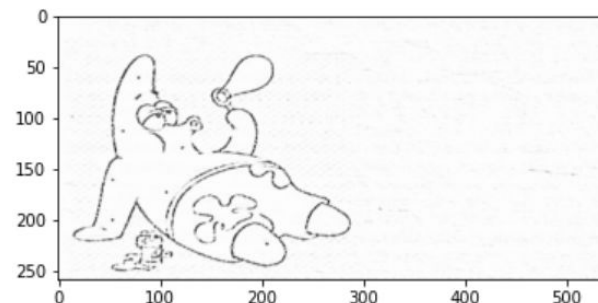
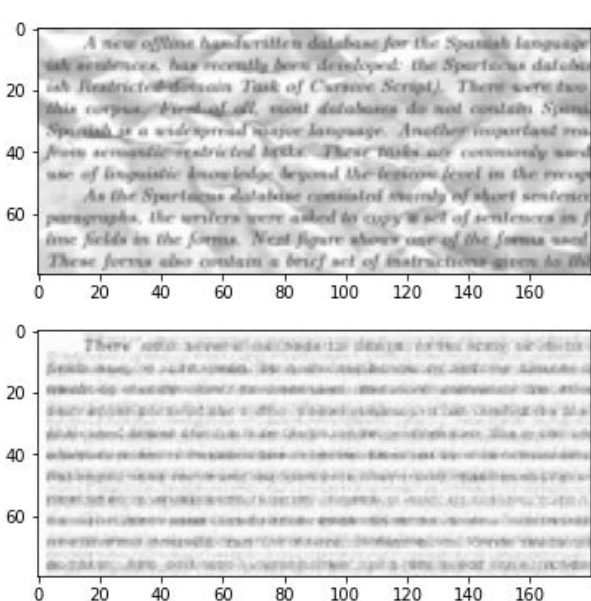


Imagen después  
del segundo  
process:

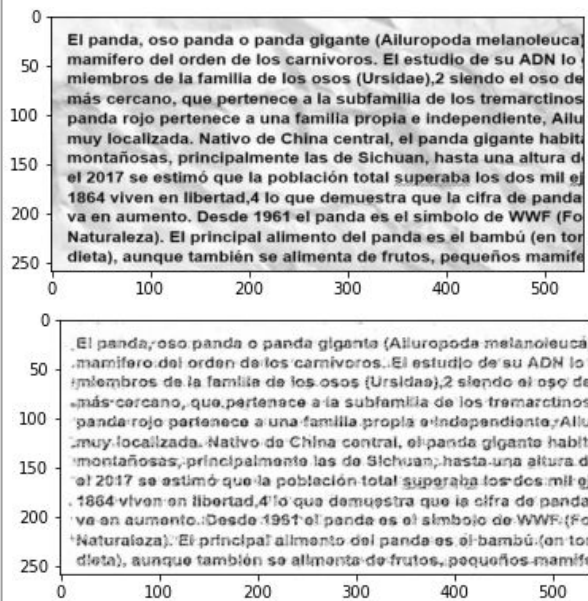


# Cuadro comparativo

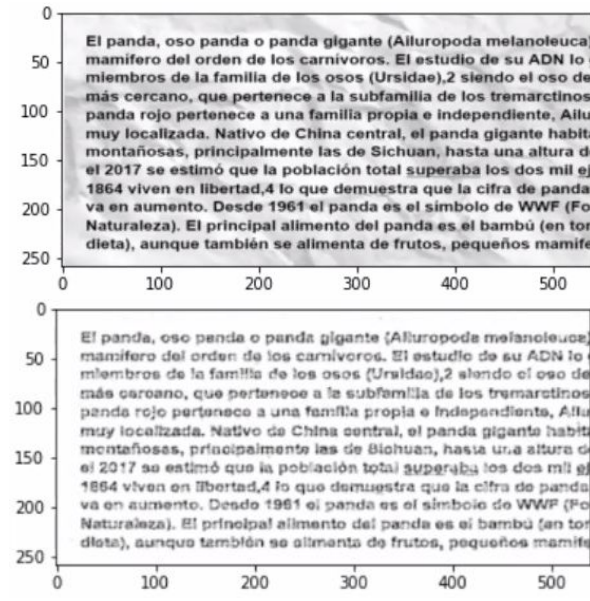
## MLP




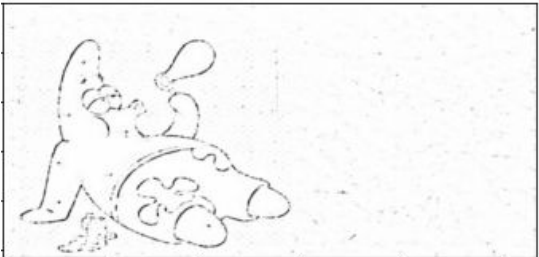

## CNN+ Process



## CNN + 2 Process



# Cuadro comparativo

Imagen original	Autoencoder 2: CNN + P	Autoencoder 3: CNN + 2P
		

# Notas

- En el caso de los encoder, la reducción de dimensionalidad influyó fuertemente en el tiempo de entrenamiento y resultados. Igualmente, se observó que las mejores opciones fueron una reducción de dimensionalidad extrema, sólo diez dimensiones para la salida del encoder, o una reducción modesta, reducir un 30% las dimensiones para la salida del encoder.
- En el caso de los decoder, el error final de predicción también es decisivo para los resultados, pues es común que la red neuronal se estanque en un valor y no alcance su mejor desempeño, esta situación es fácilmente reconocible pues las predicciones de la red neuronal serán siempre una imagen aparentemente blancas sin ningún texto. Esta condición puede ser aminorada con la selección de un adecuado learning rate.

# Recomendaciones

- Para mejorar los modelos de autoencoders se invita a explorar otros tipos de funciones de error que den mayor peso a las pequeñas diferencias entre predicciones y salidas esperadas. Pues, como se mencionó anteriormente, la red neuronal puede verse estancada y producir imágenes vacías de fondo blanco que tienen mucha similitud con la salidas esperadas pues estas también contienen un enorme área de color blanco.