Universidad de Las Palmas de Gran Canaria

Grado en Ciencia e Ingeniería de Datos

Aprendizaje Automático II

Autor:
Javier Orlando García Suárez

Enero 2024

Índice

1. Introducción

Este proyecto está formado en dos partes: la primera es una creación de una red neuronal y la segunda es un supresor de ruido de imágenes, además de realizar una super resolución.

2. Red neuronal

Para la red neuronal hemos utilizado un datasets de rostros reales y falsos, y nuestra red será una convolucional capaz de identificar cuales de estos rostros son verdaderos y cuales falsos. Hemos probado con diversos optmizadores, funciones de pérdida, data augmentation, etc. además de utilidades como early stopping para garantizar el buen funcionamiento de la red.

```
[ ]  class FaceDetection_CNN(nn.Module):
         def __init__(self):
             super(FaceDetection_CNN, self).__init__()

             self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
             self.bn1 = nn.BatchNorm2d(32)
             self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
             self.bn2 = nn.BatchNorm2d(64)

             self.pool = nn.MaxPool2d(2, 2)

             self.fc1 = nn.Linear(64*32*32, 128)

             self.fc2 = nn.Linear(128, 2)

             self.dropout = nn.Dropout(0.5)

         def forward(self, x):
             #x = self.pool(F.relu(self.conv1(x)))
             #x = self.pool(F.relu(self.conv2(x)))
             #x = self.pool(F.relu(self.conv3(x)))
             x = self.pool(F.relu(self.bn1(self.conv1(x))))
             x = self.pool(F.relu(self.bn2(self.conv2(x))))
             x = x.view(-1,64*32*32)
             x = self.dropout(F.relu(self.fc1(x)))
             x = self.fc2(x)
             return x
```

Figura 1: Red neuronal

3. Supresor de ruido

Un supresor de ruido, también conocido como filtro de reducción de ruido, es un método o algoritmo diseñado para eliminar o reducir la presencia de ruido en una señal o conjunto de datos, en este caso, hemos utilizado redes neuronales,tanto lineales comoconvolucionales para aprender patrones de ruido y suprimirlos en las imágenes.

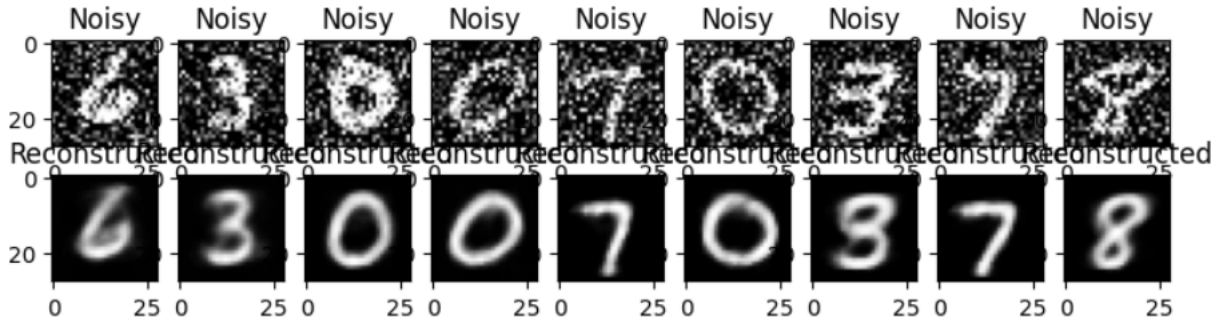

Figura 2: Resultado del supresor

4. Super resolución

Para la super resolución, primero debemos reducir las dimensiones de la imagen(encoder) para extraer características significativas y comprimir la imagen de baja resolución en un espacio de características de menor dimensión, en este caso, se realiza utilizando stride en las capas convolucionales, posteriormente, debemos aumentar las dimensiones (decider) para usar esta representación comprimida con el fin de generar una versión de alta resolución de la imagen original, para ello, usamos convoluciones traspuestas.

```python
class ImprovedSuperResolutionAutoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=3, stride=2, padding=1),  # Entrada: 1 canal, Salida: 64 canales, reduciendo las dimensio
            nn.ReLU(),
            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),  # Salida: 128 canales, reduciendo las dimensiones
            nn.ReLU(),
            nn.Conv2d(128, 256, kernel_size=3, stride=2, padding=1),  # Salida: 256 canales, reduciendo las dimensiones
            nn.ReLU()
        )

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1),  # Salida: 128 canales, aumentando las dimensiones
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1),  # Salida: 64 canales, aumentando las dimensiones
            nn.ReLU(),
            nn.ConvTranspose2d(64, 1, kernel_size=3, stride=1, padding=1),  # Salida: 1 canal
            nn.Sigmoid()
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded

# Cambios en la función de pérdida y optimizador
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(super_resolution_model.parameters(), lr=1e-3, weight_decay=1e-4)  # Agregamos weight_decay para regul
```

Figura 3: Encoder y decoder para la super resolución