Universidad del Valle de Guatemala
Facultad de Ingeniería

Deep Learning
Laboratorio 5
Reporte

Mariana David 201055
Pablo Escobar 20936

Guatemala 01 de octubre del 2023

Estructura de G(x) (Generador)

El generador en una GAN es responsable de crear muestras que se asemejen a los datos de entrenamiento. En el código que implementamos, la estructura de G(x) se define de la siguiente manera:

```python
# Definir el generador (G)
def build_generator():
    model = Sequential()
    model.add(Dense(256, input_dim=100))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(784, activation='tanh'))
    model.add(Reshape((28, 28, 1)))
    return model
```

Imagen 1. Definición del generador (G)

1. Capa de Entrada (Input Layer): la red generativa comienza con una capa de entrada que recibe vectores de ruido aleatorio de 100 dimensiones. Esto permite que el generador cree variedad en las imágenes generadas a partir de diferentes vectores de ruido.

2. Capas Densas (Dense Layers): se utilizan tres capas densas (fully connected) con 256, 512 y 1024 neuronas respectivamente. Estas capas son responsables de aumentar gradualmente la complejidad de la representación latente aprendida a partir del ruido de entrada.

3. Función de Activación LeakyReLU: después de cada capa densa, se aplica una función de activación LeakyReLU. Esta función ayuda a evitar problemas de desvanecimiento de gradientes y permite que la red aprenda representaciones no lineales.

4. Capas de Normalización por Lotes (Batch Normalization): después de cada LeakyReLU, se aplica Batch Normalization. Esto ayuda a estabilizar y acelerar el proceso de entrenamiento al normalizar las activaciones de cada capa.

5. Capa Densa de Salida: la última capa densa tiene 784 neuronas y utiliza la función de activación ”tanh”. Esta capa produce la salida del generador, que se interpreta como una imagen de 28x28 píxeles en escala de grises.

6. Capa de Reconfiguración (Reshape Layer): se agrega una capa de reconfiguración para darle forma a la salida de la capa densa en un formato de imagen 28x28x1, que es el formato de las imágenes MNIST.

Justificación de la Estructura de G(x)

La estructura del generador se ha diseñado con capas densas progresivas, lo que permite que la red aprenda representaciones cada vez más complejas a medida que profundiza en la red. Esto es fundamental para generar imágenes de calidad. Además, el uso de LeakyReLU y Batch Normalization ayuda a mitigar problemas de gradientes desvanecientes y acelera el entrenamiento. Por último, la función de activación “tanh” en la capa de salida produce valores en el rango [-1, 1], lo que se ajusta al rango de valores de las imágenes MNIST preprocesadas.

Reporte sobre la Estructura de D(x) (Discriminador)

El discriminador en una GAN es responsable de distinguir entre muestras reales y generadas por el generador. En el código que implementamos, la estructura de D(x) se define de la siguiente manera:

```python
# Definir el discriminador (D)
def build_discriminator():
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28, 1)))
    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1, activation='sigmoid'))
    return model
```

✓ 0.0s

Figura 2. Estructura de D(x) (Discriminador)

1. Capa de Entrada (Input Layer): el discriminador comienza con una capa de entrada que recibe imágenes de 28x28 píxeles en escala de grises.

2. Capa de Aplanado (Flatten Layer): La imagen de entrada se aplana en un vector unidimensional antes de ser procesada por capas densas.

3. Capas Densas (Dense Layers): el discriminador utiliza tres capas densas con 1024, 512 y 256 neuronas respectivamente. Al igual que en el generador, estas capas aumentan gradualmente la complejidad de la representación aprendida.

4. Función de Activación LeakyReLU: luego de cada capa densa, se aplica LeakyReLU para introducir no linealidad en el modelo.

5. Capa de Salida Densa: la última capa densa tiene una sola neurona y utiliza la función de activación 'sigmoid'. Esta capa produce una salida binaria que representa la probabilidad de que la entrada sea una imagen real (1) o generada (0).

Justificación de la Estructura de D(x)

La estructura del discriminador se ha diseñado con capas densas progresivas, al igual que el generador. Esto permite que el discriminador aprenda características discriminativas cada vez más complejas. Además, la función de activación LeakyReLU se utiliza para evitar problemas de gradientes desvanecientes y mejorar la capacidad del discriminador para aprender representaciones no lineales. Por último, la capa de salida utiliza la función de activación 'sigmoid', que es adecuada para la tarea de clasificación binaria entre imágenes reales y generadas.

Comparación de resultados antes y después de su proceso de entrenamiento

Figura 3. Resultados sin entrenar

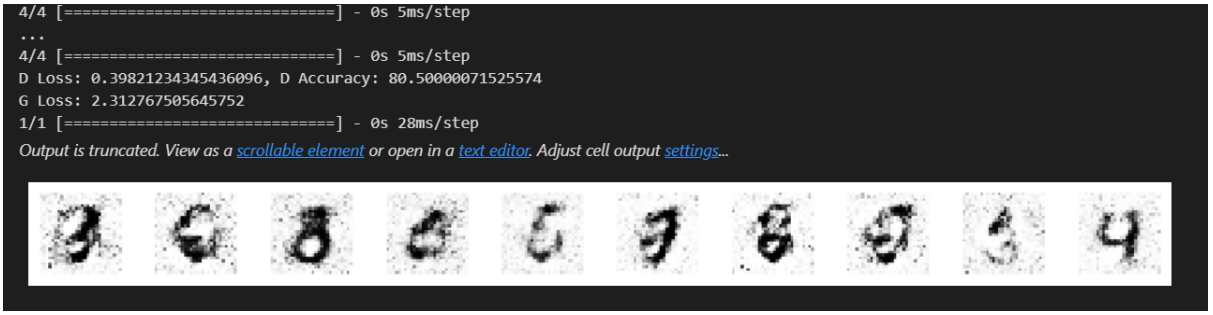

Figura 4. Resultados entrenados (epoch:2-size:100)

Conclusión

Las estructuras de G(x) y D(x) se han diseñado cuidadosamente para permitir que el generador y el discriminador aprendan y generen imágenes realistas de acuerdo con el conjunto de datos MNIST. Estas estructuras son fundamentales para el funcionamiento de la GAN y han sido elegidas para equilibrar la capacidad de generación y discriminación de la red. Además como vemos en los resultados finales, luego de ser entrenado el cambio es bastante notorio. La definición del resultado si entrenado no es completamente buena pero si identificable en su mayoría, lo cual indica que con un entrenamiento más exhaustivo los resultados serán mucho mejores.