Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz

Trabajando con AutoEncoders y Conceptos de Redes Adversarias Generativas

Universidad Autónoma de Occidente

Redes Neuronales Artificiales y Deep Learning

Melina González - 2236748
melina.gonzalez@uao.edu.co
Juan Camilo León - 2190612
juan_camilo.leon@uao.edu.co
Juan Camilo Caicedo Cortes – 2190732
juan_c.caicedo@uao.edu.co

1. **Realice una investigación de dos posibles aplicaciones donde se usen Variational Autoencoders (VAE) para la generación de algún tipo información.**

El ejercicio 1 se detalla a continuación:

Los Variational Autoencoders (VAEs) son un tipo de modelos gráficos probabilísticos basados en redes neuronales que manejan conjuntos de datos de gran tamaño.

Los VAEs tienen la capacidad de generar nuevos datos que se asemejan a aquellos que pertenecen a la distribución original. Esta capacidad se logra mediante la codificación de las características fundamentales de la distribución de probabilidad representada por la población de datos originales en forma de variables latentes. A partir del modelo que han aprendido, los VAEs son capaces de generar nuevas instancias que se acercan a dicha distribución.

Los VAEs son poderosos modelos generativos, que en los últimos tiempos están encontrando aplicaciones en campos muy diversos que van desde generar rostros humanos hasta a producir muestras de audio sintético.

A continuación, se presentan dos aplicaciones comunes donde se utilizan VAE para la generación de información:

- **Generación de Rostros y Avatares:** En aplicaciones de realidad virtual, videoconferencia y juegos en línea, los VAE se utilizan para generar avatares personalizados y realistas a partir de una foto de una persona.
- **Generación de Datos de Sensor y Robótica:** En robótica y aplicaciones de sensores, los VAE se utilizan para generar datos de sensores sintéticos que ayudan en el entrenamiento de algoritmos de control y planificación.

2. **Realice una investigación de dos posibles aplicaciones donde se usen Redes Adversarias Generativas (GAN) para la generación de algún tipo información. ¿Cuál es la diferencia entre un VAE y una GAN cuando se aplica para la generación de información?**

El ejercicio 2 se detalla a continuación:

Las Redes Adversarias Generativas (GANs) son un enfoque que implica una competencia entre dos redes neuronales: una llamada generadora y la otra discriminadora. El propósito principal es generar

nuevas instancias que sean prácticamente indistinguibles de las que se encuentran en la distribución de probabilidad a partir de la cual se originan los datos utilizados para el entrenamiento.

La base teórica subyacente permite que estas GANs sean aplicables a la generación de diversos tipos de datos. Se ha comprobado su efectividad en campos tan variados como la visión por computadora, la segmentación semántica, la síntesis de series temporales, la edición de imágenes, el procesamiento del lenguaje natural y la generación de imágenes a partir de texto, entre otros.

A continuación, se presentan dos aplicaciones comunes donde se utilizan GANs para la generación de información:

- **Restauración y Mejora de Imágenes:** Las GAN se utilizan para restaurar imágenes antiguas, eliminar el ruido de las fotos y mejorar la calidad de las imágenes, lo que es valioso en la fotografía y la restauración de documentos históricos.
- **Animación y Videojuegos:** Las GAN se utilizan para generar personajes y escenarios en animaciones y videojuegos, permitiendo la creación de mundos virtuales convincentes y animaciones de alta calidad.

Diferenciación:

La diferencia entre estos dos enfoques radica en que, los VAEs tienen la capacidad de generar nuevos datos que se asemejan a aquellos que pertenecen a la distribución original, mientras que las Redes Adversarias Generativas (GANs) son un enfoque que implica una competencia entre dos redes neuronales: una llamada generadora y la otra discriminadora.

Otra de las diferencias notables, es con respecto a la calidad, puesto que los VAEs tienden a generar datos más diversificados, ya que el espacio latente es continuo y permite la interpolación suave entre instancias, mientras que las GAN pueden generar datos de alta calidad, pero a veces pueden tener dificultades para mantener la diversidad en las muestras generadas.

Por último, se tiene que, de acuerdo al entrenamiento y estabilidad, Los VAE son más estables durante el entrenamiento, pero pueden generar datos menos realistas en comparación con las GAN. Por otro lado, se tiene que las GAN pueden ser más difíciles de entrenar y pueden sufrir de problemas como el colapso del modo y la generación de artefactos, pero son conocidas por su capacidad para producir datos visuales de alta calidad.

3. **Realice una aplicación de clasificación utilizando Autoencoders apilados usando la base de datos MNIST.**
4. **Entrene un Autoencoder para visualizar en dos dimensiones el comportamiento de la base de datos Fashion MNIST. Realice una visualización similar usando PCA (Análisis de Componentes Principales) y compare los resultados obtenidos. Adicionalmente, verifique la capacidad de generar nuevas imágenes con el Autoencoder entrenado.**
5. **Realice una aplicación de clasificación utilizando un Autoencoders apilados con la base de datos Fashion MNIST**

Los ejercicios 3, 4 y 5 se detallan a continuación:

Fashion MNIST es un dataset con imágenes en blanco y negro de diez clases de ropa y calzados. Estas imágenes están etiquetadas y se encuentran separadas en dos conjuntos, uno de entrenamiento (con 60000 imágenes) y uno de prueba o validación (con 10000 imágenes. Los datos se encuentran en un archivo con extensión .csv o bien en un formato que contiene los bytes de datos.

Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz

En el enlace https://colab.research.google.com/drive/1JFEyXdPi3e9PCN9HtDE_vPOcl1FJYZvP?usp=sharing se encuentra el modelo de la red neuronal del Autoencoder creado para visualizar los datos usando PCA, TSNE y el mismo modelo del Autoencoder.

"*El análisis de componentes principales (PCA) es un método para reducir la dimensionalidad de los datos y se utiliza para mejorar la visualización de datos y acelerar el entrenamiento de modelos de aprendizaje automático.*" [1]

En cambio, "*La incrustación de vecinos [estocásticos] distribuida en T (t-SNE) es una técnica de reducción de dimensionalidad que ayuda a los usuarios a visualizar conjuntos de datos de alta dimensión. Toma los datos originales que se ingresan en el algoritmo y hace coincidir ambas distribuciones para determinar cómo representar mejor estos datos usando menos dimensiones.* " [2]

Primero, antes de poder realizar este análisis, debe crearse la arquitectura del autoencoder y entrenarse posteriormente. El mismo está conformado por tres partes principales: Decoder, latent layers y encoder. El decoder tiene una arquitectura conformada como se muestra a continuación:

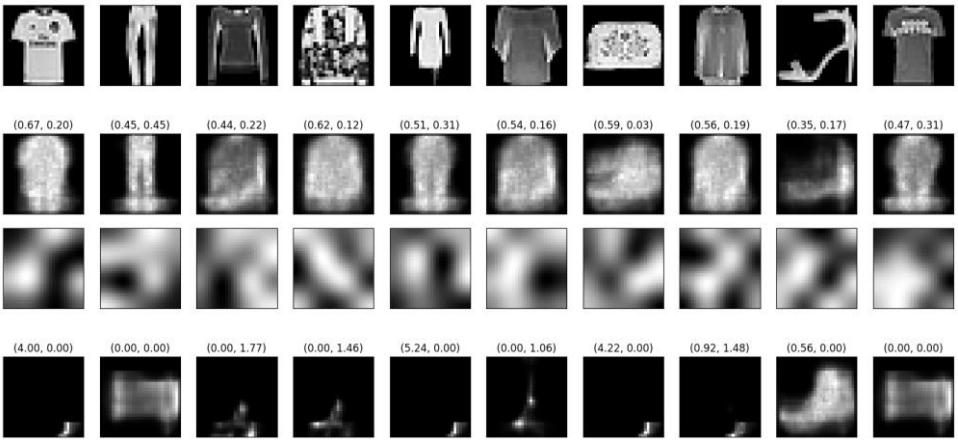
```
inputs = Input(shape=(28, 28, 1))

x = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = BatchNormalization()(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(32, (2, 2), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2))(x)
x = BatchNormalization()(x)
x = Conv2D(16, (2, 2), activation='relu', padding='same')(x)
x = Conv2D(4, (2, 2), activation='relu', padding='same')(x)
x = Conv2D(1, (2, 2), activation='relu', padding='same')(x)
x = Flatten()(x)
encoded = Dense(2, activation='relu')(x)

encoder = Model(inputs=inputs, outputs=encoded)
```

El encoder, en cambio, produce el efecto opuesto al decoder, y su programación es la siguiente:

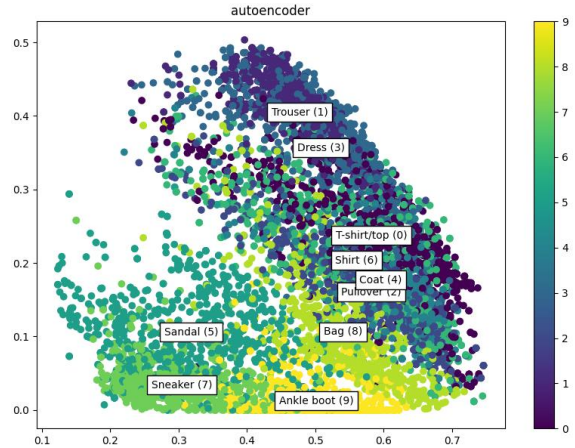
```
x = Dense(4, activation='relu')(encoded_inputs)
x = Reshape((2, 2, 1))(x)
x = Conv2D(4, (2, 2), activation='relu', padding='same')(x)
x = Conv2D(16, (2, 2), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = UpSampling2D((7, 7))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

decoder = Model(inputs=encoded_inputs, outputs=decoded)
```

La siguiente es una representación de lo entregado por el decoder luego de ser entrenado:

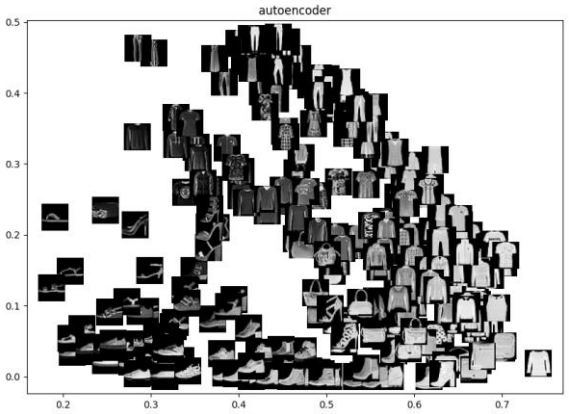
Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz



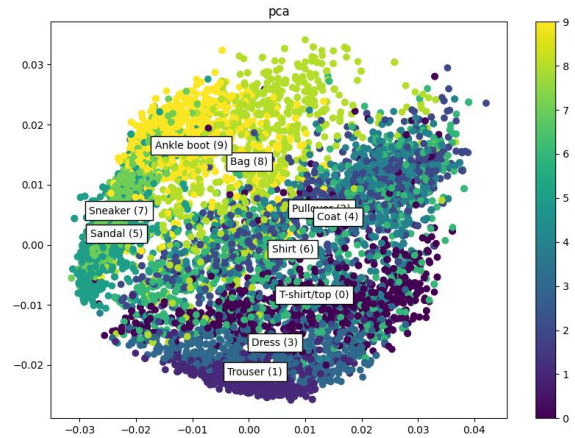
Entre el decoder y el encoder existen las capas latentes, que fueron creadas con una función. A su vez esta permite realizar los gráficos de visualización de los datos con los distintos métodos implementados, los cuales quedan como se ve en las siguientes imágenes:



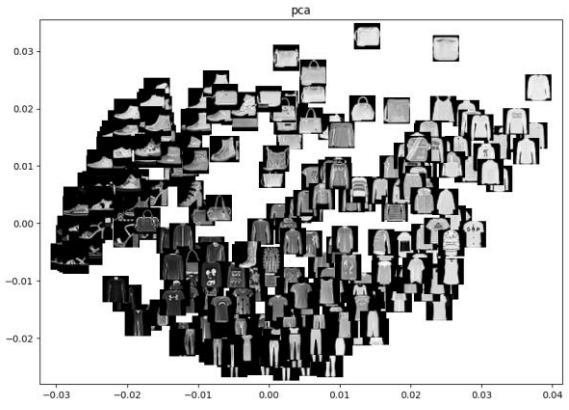
Visualización de datos con nube de puntos con Autoencoder



Visualización de datos con nube de imágenes con Autoencoder

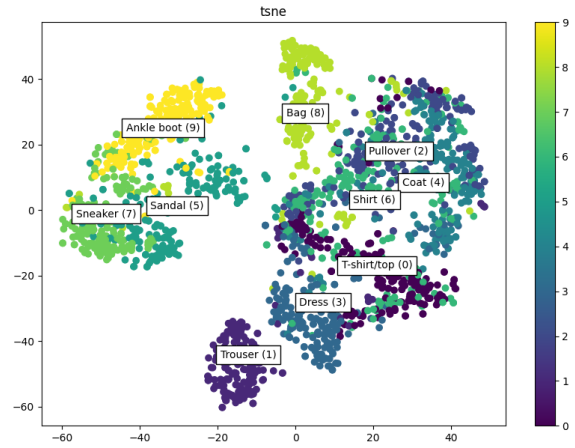


Visualización de datos con nube de puntos con PCA

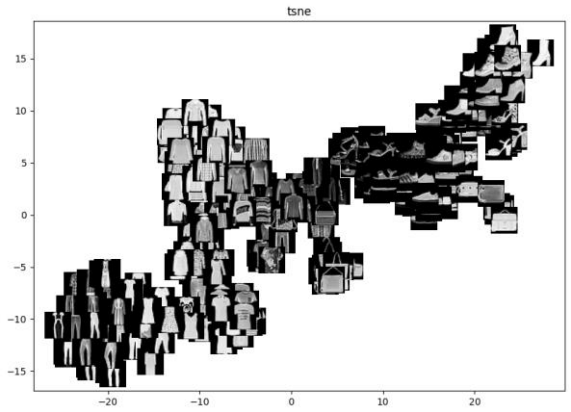


Visualización de datos con nube de imágenes con PCA

Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz



Visualización de datos con nube de puntos con TSNE



Visualización de datos con nube de imágenes con TSNE

Comparando las técnicas comunes de reducción de dimensionalidad: PCA y T-SNE, se reduce el espacio de 784 dimensiones de las imágenes a dos usando la técnica que se quiere. Luego se traza la distribución de las diez clases en el nuevo espacio bidimensional y verá qué tan separadas están. Esto es lo que se realizó en las imágenes anteriores.

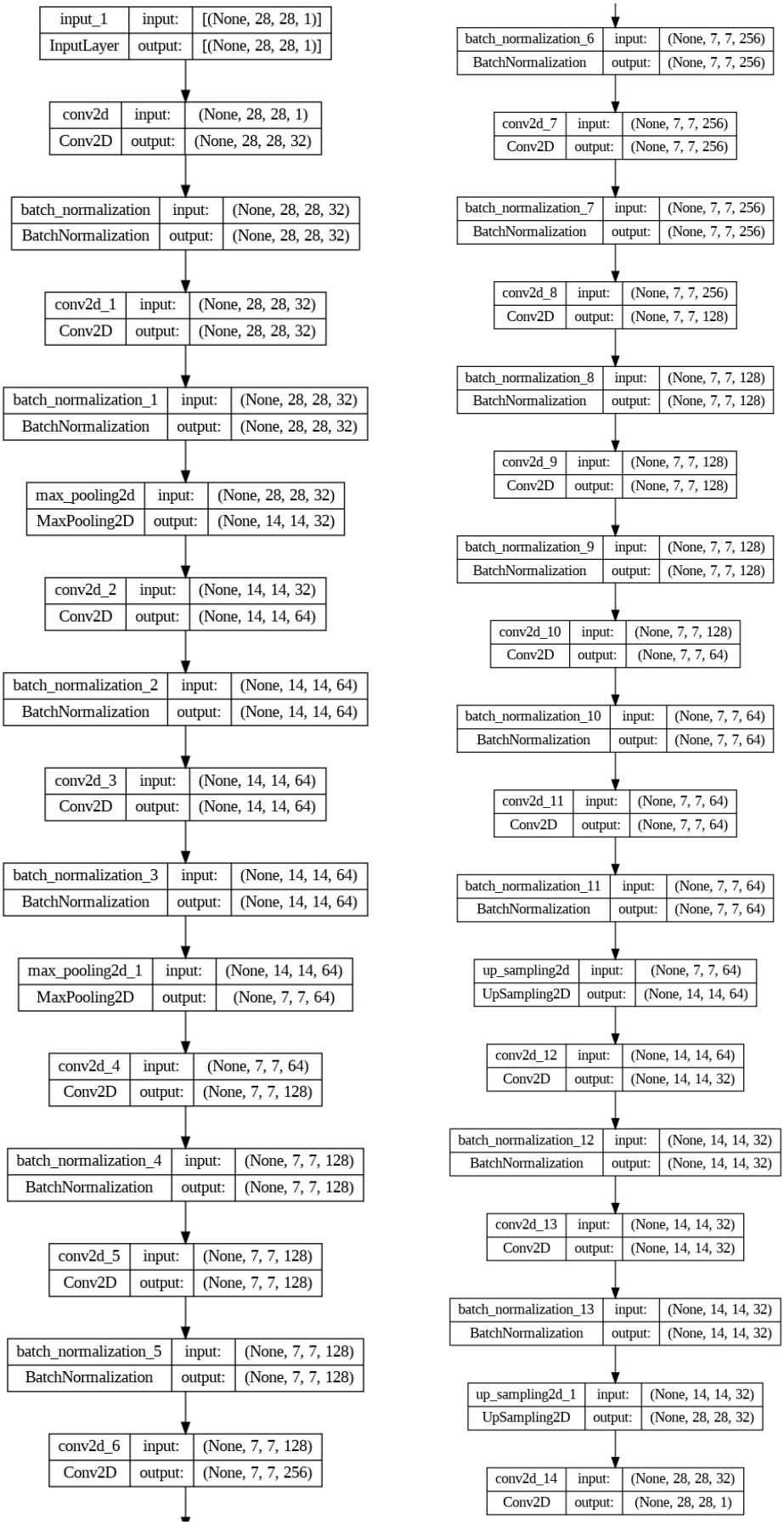
Con Autoencoder los datos se visualizan en una única línea, independientemente de su clase, mientras que con PCA se crea una nube de puntos que se aglomera de acuerdo a los valores de los datos, en este caso de los pixeles de las imágenes. Como se ve en la Visualización de datos con nube de imágenes con PCA, a la derecha se encuentran las imágenes con mayor cantidad de color blanco mientras que a la izquierda tienen mayor cantidad de negro. En cambio, con la técnica de TSNE los datos se agrupan según sus semejanzas y a su vez bien delimitados por clase. Por eso aparecen “juntos” todos los calzados por un lado, por otro remeras, vestidos, camperas, y por otro pantalones. Esta es la agrupación por “vecinos”, porque las imágenes se parecen en la morfología de su contenido.

---o---o---o---

En el enlace https://colab.research.google.com/drive/1W9d7PBANzStaclDTNY-SbV5yyT0SEjSS?usp=sharing se encuentra el modelo utilizado para verificar la capacidad de generar nuevas imágenes con el Autoencoder entrenado y la clasificación de las imágenes del dataset usando un Autoencoder.

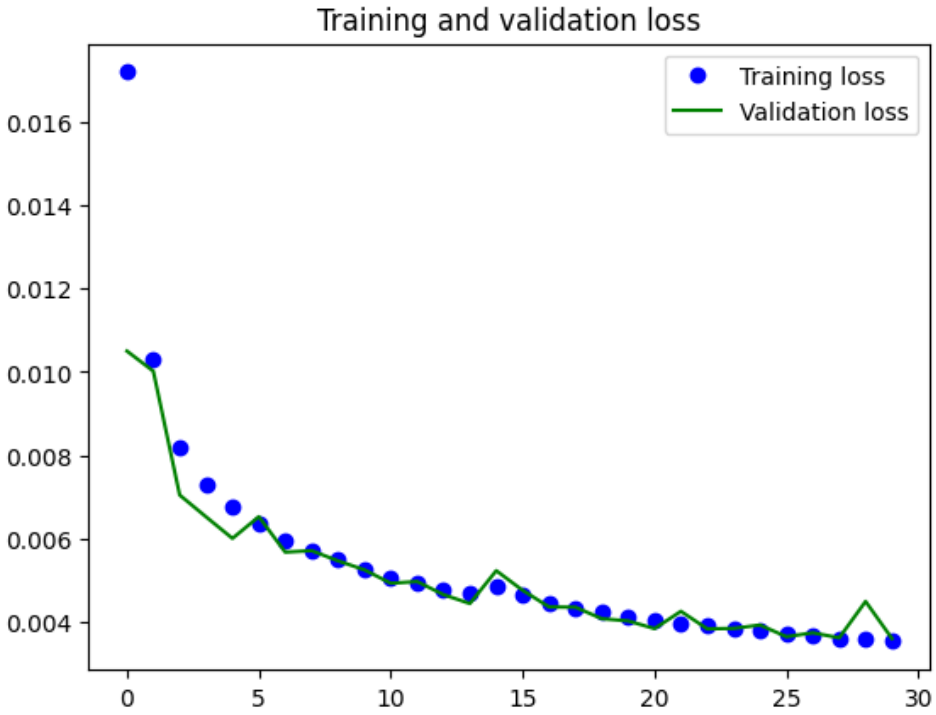
La arquitectura del Autoencoder se visualiza en la imagen siguiente (leer primero a la izquierda y luego a la derecha).

Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz



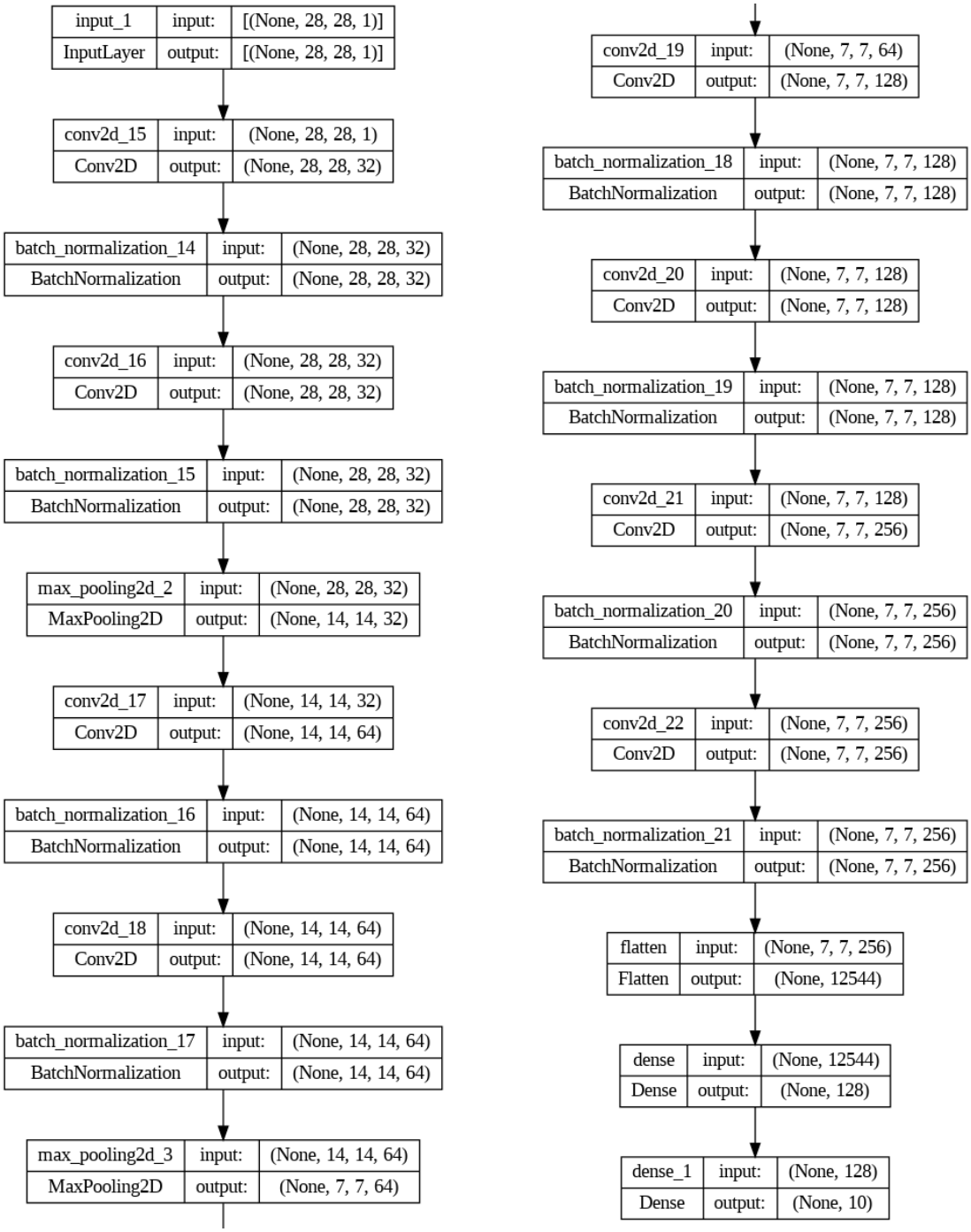
Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz

En el caso del entrenamiento con Autoencoder se obtuvieron los siguientes valores de la función de pérdida, tanto en el caso del entrenamiento como en la validación del mismo. Como se observa, estos valores son coherentes entre sí y las curvas son semejantes.



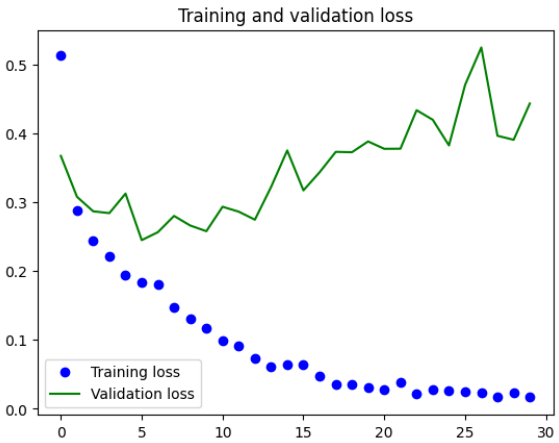
La arquitectura del modelo completo es la siguiente: (leer primero a la izquierda y luego a la derecha).

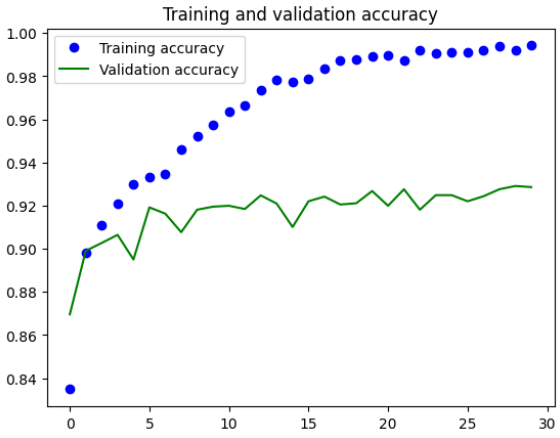
Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz



En este caso los valores de pérdida y precisión son los mostrados en las siguientes imágenes, respectivamente. Puede observarse que la pérdida en el caso de los datos de validación se ve incrementada, mientras que la de los datos de entrenamiento disminuye. Esto puede ocurrir debido a un sobreentrenamiento de la red. En concordancia con lo dicho anteriormente, la función de precisión de los datos de validación tiene un pequeño crecimiento pero luego se mantiene constante en promedio, mientras que la curva de precisión de los datos de entrenamiento aumenta considerablemente. Esto confirma el sobreentrenamiento, dado que es posible que la red se haya aprendido los datos de entrenamiento.

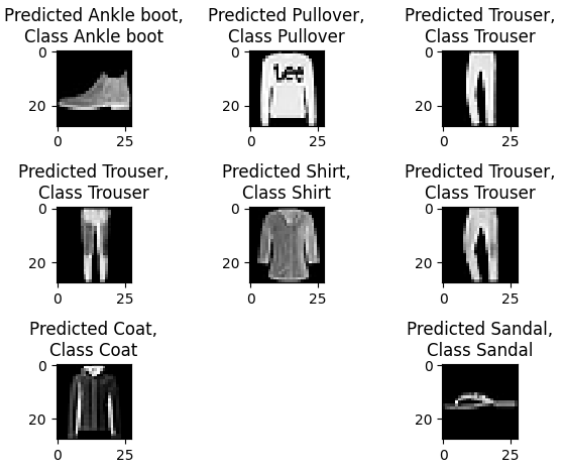
Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz



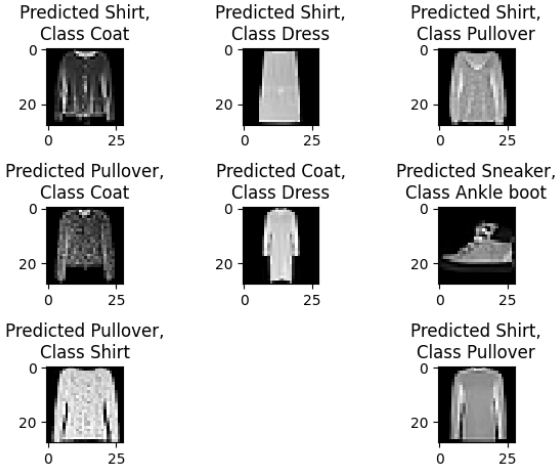


Luego se entrenó la red y se testeó este entrenamiento. Hubo casos en los que las predicciones de las imágenes fueron correctas pero en otros casos esta predicción no fue exitosa. A continuación se muestran algunos ejemplos:

Clasificación exitosa



Clasificación con error



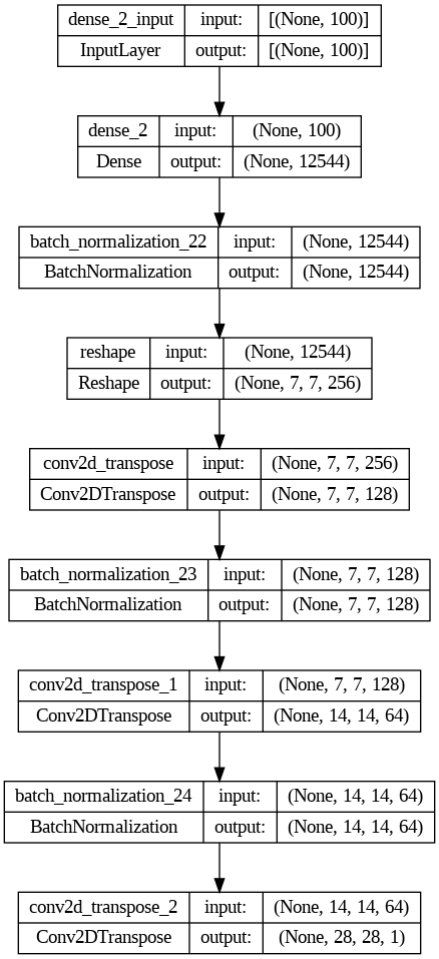
Para poder evaluar esta clasificación es que se presenta un reporte de la misma para cada clase:

	precision	recall	f1-score	support
Class 0	0.85	0.89	0.87	1000
Class 1	1.00	0.99	0.99	1000
Class 2	0.89	0.88	0.89	1000
Class 3	0.94	0.92	0.93	1000
Class 4	0.88	0.89	0.89	1000
Class 5	0.99	0.97	0.98	1000
Class 6	0.79	0.77	0.78	1000
Class 7	0.96	0.98	0.97	1000
Class 8	0.99	0.99	0.99	1000
Class 9	0.97	0.97	0.97	1000
accuracy			0.93	10000
macro avg	0.93	0.93	0.93	10000
weighted avg	0.93	0.93	0.93	10000

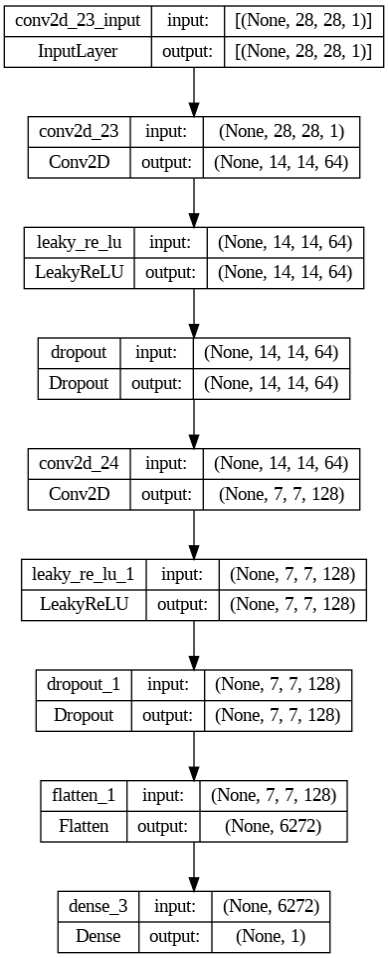
Para el modelo generativo se tienen las siguientes arquitecturas:

Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz

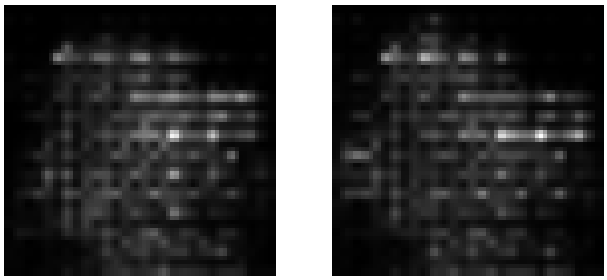
Arquitectura del Generador



Arquitectura del Discriminador



Luego, fue capaz de generar las siguientes dos imágenes luego de 300 épocas:



Como se observa, las imágenes generadas aún no tienen una forma definida. Ello implica que debe reentrenarse la misma con más épocas o utilizando una arquitectura diferente de modo que a la salida se obtenga una imagen con una forma semejante a las imágenes pertenecientes al dataset. Para ello debe tenerse en cuenta los recursos computacionales necesarios para llevar a cabo esta tarea y el tiempo que demanda. Para entrenar 300 épocas utilizando una T4 GPU proporcionada por Google, el entrenamiento duró aproximadamente tres horas.

Bibliografía consultada:

Melina Paula Gonzalez Rubiño
Juan Camilo León López
Juan Camilo Caicedo Cortéz

[1] M. Galarnyk. “PCA Using Python: A Tutorial”. Built In. Accedido el 3 de noviembre de 2023. [En línea]. Disponible: https://builtin.com/machine-learning/pca-in-python

[2] L. Derksen. “Using T-SNE in Python to Visualize High-Dimensional Data Sets”. Built In. Accedido el 3 de noviembre de 2023. [En línea]. Disponible: https://builtin.com/data-science/tsne-python

[3] De la Torre, J. (2023a). *AUTOCODIFICADORES VARIACIONALES (VAE) FUNDAMENTOS TEÓRICOS Y APLICACIONES* (Ph.D. in Computer Science (ML/AI)). https://arxiv.org/ftp/arxiv/papers/2302/2302.09363.pdf

[4] De la Torre, J. (2023b). *REDES GENERATIVAS ADVERSARIAS (GAN) FUNDAMENTOS TEÓRICOS Y APLICACIONES* (Universitat Oberta de Catalunya). https://arxiv.org/ftp/arxiv/papers/2302/2302.09346.pdf