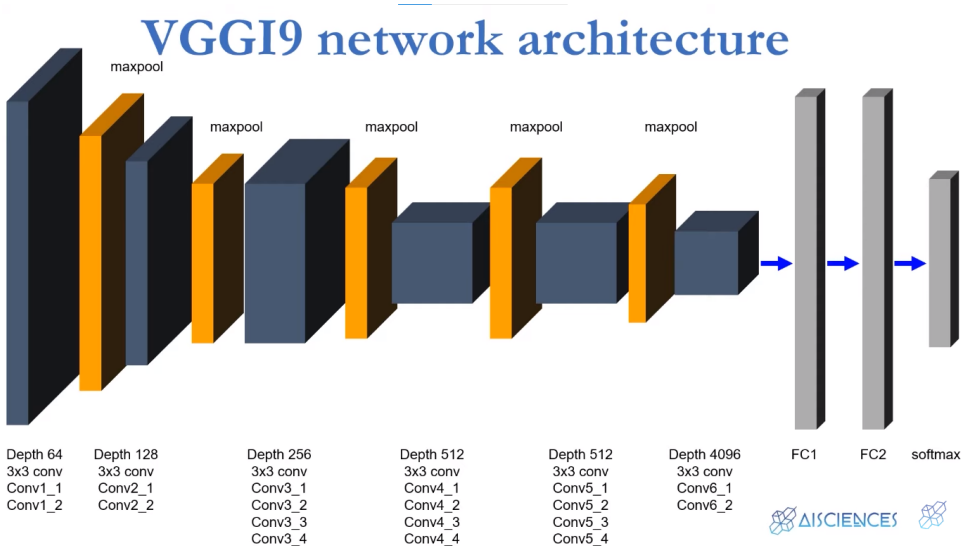
Modelo VGG19:

Un código VGG19 no se refiere a un programa de computadora en sí, sino a una arquitectura específica de red neuronal convolucional (CNN) llamada "VGG19". VGG19 es un modelo de red neuronal que fue desarrollado por el grupo Visual Geometry Group (VGG) en la Universidad de Oxford. El nombre "VGG19" se deriva de su estructura, que consta de 19 capas, incluyendo capas convolucionales y completamente conectadas.

VGG19 es conocido por su simplicidad y profundidad en comparación con otras arquitecturas de redes neuronales profundas. Todas las capas convolucionales en VGG19 utilizan filtros de 3x3, y las capas de agrupamiento máximo se utilizan para reducir progresivamente la resolución espacial de las características extraídas. VGG19 ha demostrado ser efectivo en tareas de visión por computadora, como la clasificación de imágenes, la detección de objetos y la segmentación semántica.

Un código VGG19 generalmente se refiere a la implementación de la arquitectura VGG19 en un lenguaje de programación específico, como Python, utilizando una biblioteca de aprendizaje profundo como TensorFlow o PyTorch. Este código incluiría la definición de las capas y parámetros de la red, así como las operaciones de entrenamiento y predicción. Un código VGG19 puede ser utilizado para construir y entrenar un modelo VGG19 personalizado en una tarea específica o para realizar transferencia de aprendizaje utilizando un modelo preentrenado en conjuntos de datos más grandes, como ImageNet.



Modelo LinearSVM:

Entendiendo el Modelo de Máquinas de Soporte Vectorial (SVM) en la Clasificación de Imágenes utilizando Scikit-Learn

La clasificación de imágenes es una tarea fundamental en el campo de la visión por computadora y el aprendizaje automático. La capacidad de identificar y etiquetar imágenes automáticamente tiene una amplia gama de aplicaciones, desde el reconocimiento facial hasta la detección de objetos y la clasificación de contenido. Entre las numerosas técnicas de clasificación de imágenes, una de las más poderosas y versátiles es el modelo de Máquinas de Soporte Vectorial (SVM). En este artículo, exploraremos qué es un modelo SVM y cómo implementarlo en la clasificación de imágenes utilizando Scikit-Learn, una biblioteca de aprendizaje automático en Python.

Contenido:

1. Introducción a las Máquinas de Soporte Vectorial (SVM)

Las Máquinas de Soporte Vectorial (SVM) son un conjunto de algoritmos de aprendizaje supervisado que se utilizan para la clasificación y regresión. Las SVM se destacan en la clasificación de datos cuando las clases son linealmente separables, es decir, se pueden

separar de manera efectiva mediante una línea recta o un hiperplano. Sin embargo, las SVM también pueden manejar datos no linealmente separables utilizando funciones de base (kernels) que transforman los datos a un espacio de mayor dimensión donde la separación lineal es posible.

En el contexto de la clasificación de imágenes, las SVM son conocidas por su capacidad para manejar conjuntos de datos de alta dimensión y la flexibilidad para trabajar con diferentes tipos de datos, incluyendo imágenes. La idea principal detrás de SVM es encontrar un hiperplano que maximice la separación entre las clases. En el caso de la clasificación binaria, esto significa encontrar el hiperplano que separa mejor dos clases, mientras que en la clasificación multiclase, se busca encontrar varios hiperplanos que distingan cada clase de las demás.

Funcionamiento de las SVM

El funcionamiento de las SVM se puede resumir en los siguientes pasos:

1- Mapeo de Datos a un Espacio de Características de Mayor Dimensión: Si los datos no son linealmente separables en el espacio original, se pueden mapear a un espacio de características de mayor dimensión utilizando funciones de mapeo (kernels) para hacer que sean linealmente separables en ese espacio.
2- Encontrar el Hiperplano Óptimo: Las SVM buscan el hiperplano de decisión óptimo que separe las clases y maximice el margen. El hiperplano se selecciona de tal manera que sea equidistante de los puntos de datos más cercanos de ambas clases.
3- Tratamiento de Puntos Atípicos (Outliers): Las SVM son robustas frente a puntos atípicos, ya que su objetivo principal es encontrar un margen máximo y no verse influenciadas por puntos individuales.
4- Clasificación de Nuevos Datos: Una vez entrenado, el modelo SVM puede utilizarse para clasificar nuevos datos en una de las clases.

Tipos de SVM

Hay varios tipos de SVM, pero los dos más comunes son:

SVM Lineal: Se utiliza cuando se supone que los datos son linealmente separables. Busca un hiperplano que divida las clases de manera lineal.

SVM no Lineal: Se utiliza cuando los datos no son linealmente separables en el espacio original. Aplica funciones de mapeo (kernels) para llevar los datos a un espacio de características de mayor dimensión donde sean linealmente separables. Los kernels comunes incluyen el kernel polinómico y el kernel radial (RBF).

2. Clasificación de Imágenes con SVM

Clasificación de Imágenes

La clasificación de imágenes es una tarea fundamental en la visión por computadora y el procesamiento de imágenes. Implica asignar una etiqueta o categoría a una imagen en función de su contenido visual. Las aplicaciones de la clasificación de imágenes son amplias y van desde la detección de objetos en imágenes médicas hasta la identificación de objetos en imágenes de satélite.

Las SVM son una elección popular en la clasificación de imágenes debido a su capacidad para trabajar con datos de alta dimensión y su robustez frente a problemas de sobreajuste (overfitting). Los modelos SVM pueden aprender a distinguir características visuales en imágenes y realizar tareas de clasificación en función de esas características

EMPEZAMOS CON EL CODIGO:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

```
# Cargar el conjunto de datos "Digits"
digits = datasets.load_digits()
X, y = digits.data, digits.target
```

```
# Dividir el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Crear y entrenar el modelo SVM
model = SVC()
model.fit(X_train, y_train)
```

```
# Realizar predicciones en el conjunto de prueba
y_pred = model.predict(X_test)
```

```
# Calcular la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print(f'Precisión del modelo: {accuracy:.2f}')
```

```
Precisión del modelo: 0.99
```

5. Conclusiones

Las Máquinas de Vectores de Soporte (SVM) son un enfoque poderoso para la clasificación de imágenes y otras tareas de aprendizaje automático. Su capacidad para trabajar con datos de alta dimensión y su capacidad para manejar problemas de clasificación no lineal hacen que sean una elección sólida en una variedad de aplicaciones.

En este artículo, hemos explorado cómo implementar un clasificador SVM en Scikit-Learn para clasificar imágenes. Hemos utilizado el conjunto de datos "Digits" como ejemplo y hemos seguido los pasos necesarios para cargar, preparar, entrenar y evaluar el modelo SVM.

Es importante destacar que las SVM son solo uno de los muchos enfoques de clasificación de imágenes disponibles en el campo del aprendizaje automático. La elección del algoritmo adecuado depende en gran medida de la naturaleza del problema y de los datos disponibles.

ahora usamos un modelo CNN para la misma tarea:
El modelo utilizado en este código es una red neuronal de tipo "feedforward" o red neuronal multicapa (MLP, por sus siglas en inglés). Específicamente, es un modelo de clasificación de dígitos escritos a mano que se utiliza para predecir a qué número corresponde una imagen de 28x28 píxeles

El modelo funciona a la perfeccion para la clasificacion de numeros escritos a mano.

Por ultimo lugar, la base de datos de CIFAR10, que ustedes ya conocen pero lo entrenamos con un modelo VGG19.
Aquí una breve explicacion del codigo:
Importa las bibliotecas necesarias de TensorFlow y Keras para construir y entrenar el modelo, así como para cargar el conjunto de datos y aplicar aumentación de datos.

Define el nuevo tamaño al que se redimensionarán las imágenes de entrada, que es 64x64 píxeles.

Crea un generador de datos de imágenes (datagen) para aplicar aumentación de datos. Esto incluye la rescalación de los valores de píxeles, rotación, cambio de ancho y alto, inversión horizontal y un rango de zoom.

Carga el conjunto de datos CIFAR-10, que consiste en imágenes etiquetadas de 32x32 píxeles.

Redefine el tamaño de las imágenes en el conjunto de prueba (x_test) redimensionándolas a 64x64 píxeles

Normaliza las etiquetas y_train y y_test utilizando to_categorical para asegurarse de que estén en el formato adecuado para la clasificación de múltiples clases (one-hot encoding).

Redefine el tamaño de las imágenes en el conjunto de entrenamiento (x_train) redimensionándolas a 64x64 píxeles.

Crea un generador de entrenamiento (train_generator) utilizando el generador de datos datagen. Este generador se utiliza para generar lotes de imágenes de entrenamiento aumentadas y sus etiquetas a medida que se entrena el modelo.

Carga el modelo VGG19 preentrenado en ImageNet sin la capa superior. La arquitectura VGG19 es una red neuronal convolucional que ha sido entrenada en un gran conjunto de datos de imágenes para tareas de clasificación.

Congela las capas del modelo base para evitar que se actualicen durante el entrenamiento. Esto se hace estableciendo trainable en False para cada capa del modelo base.

Agrega una nueva capa de clasificación en la parte superior del modelo. Esta capa incluye una operación de aplanamiento (Flatten), seguida de una capa densa con 512 unidades y función de activación ReLU, y finalmente una capa de salida con 10 unidades y función de activación softmax para clasificar las 10 clases en el conjunto de datos CIFAR-10.

Crea el modelo final especificando las entradas (base_model.input) y las salidas (predictions).

Compila el modelo utilizando el optimizador Adam con una tasa de aprendizaje de 0.0001, la función de pérdida categorical_crossentropy (adecuada para la clasificación de múltiples clases) y la métrica de precisión.

Entrena el modelo utilizando el generador de entrenamiento (train_generator) y valida en el conjunto de prueba (x_test / 255.0) durante 10 épocas.

Evalúa el modelo en el conjunto de prueba y muestra la pérdida y la precisión.

En resumen, este código implementa una red neuronal VGG19 preentrenada en el conjunto de datos CIFAR-10 después de realizar la aumentación de datos y la preparación de los datos de entrenamiento y prueba. Luego, se entrena el modelo y se evalúa su rendimiento en la clasificación de imágenes de objetos en 10 clases diferentes. La arquitectura VGG19 preentrenada se combina con capas personalizadas para la clasificación de múltiples clases.