Deteccion y Reconocimiento Facial para Evitar Suplantacion de Identidad

Guido Anthony Chipana Calderon
Universidad Nacional de Ingenieria, Lima, Peru
guido.chipana.c@uni.pe

***Abstract*—Este proyecto se centrara en el desarrollo de un sistema de deteccion y reconocimiento facial para prevenir la suplantacion de identidad en diferentes areas donde se requiera identificarlas, utilizando tecnicas de aprendizaje aprendizaje automatico.**
La implementacion estara basada en redes neuronales convoluciones (CNN) y haciendo uso de herramientas de procesamiento de imagenes como OpenCV, y Keras para construir la CNN. El objeitvo principal es crear un sistema de autenticacion seguro, eficiente y robusto que pueda identificar la identidad de una persona a traves de sus caracteristicas faciales. Y que metodologia seguiremos para realizar con exito este proyecto?
La metodologia que propongo incluye la recoleccion de datos, el procesamientos de estos datos, la construccion y entrenamiento de la CNN.
Con esto espero que se mejore la precision en la autenticacion facial con respecto a otros modelos ya existentes, asi minimizando el riesgo suplantacion de identidad.

***Index Terms*—OpenCV, Keras, CNN**

I. Introduccion

El reconocimiento facial en los ultimos años a tomado gran relevancia en las areas de seguridad, autenticacion, etc. Esto debido a la capacidad de las maquinas para detectar y reconocer rostros humanos de manera eficiente y precisa, sin embargo aun persisten problemas en la presicion y la capacidad de adaptacion a diferentes condiciones como la luz, angulo, expresiones faciales, etc.
Sabemos que existen otros metodos de autenticacion como son las contraseñas y tarjetas de identificacion, pero el reconocimiento facial no solo mejora la seguridad para evitar suplantacion de identidad, sino que tambien agiliza el proceso de verificacion sin necesidad de acordarse de una contraseña o tener a la mano una tarjeta de identificacion.

A. Objetivos

- Desarrollar un modelo de red neuronal convolucional (CNN) capaz de detectar y reconocer rostros humanos en diversas condiciones.
- Evaluar que tan efectivo es el modelo.
- Comparar el rendimiento de mi modelo con otros modelos ya existentes.

Espero que el proyecto y modelo creado sea utilizado en la Universidad Nacional de Ingenieria del Peru, para agilizar y evitar suplantacion de identidad en practicas calificas o examenes.

II. Fundamentos

A. OpenCV

OpenCV(Open Source Computer Vision Library) es una biblioteca de codigo abierto diseñada para realizar tareas de procesamiento de imagenes y vision por computadora. En el proyecto se utilizara para preprocesar las imagenes faciales.

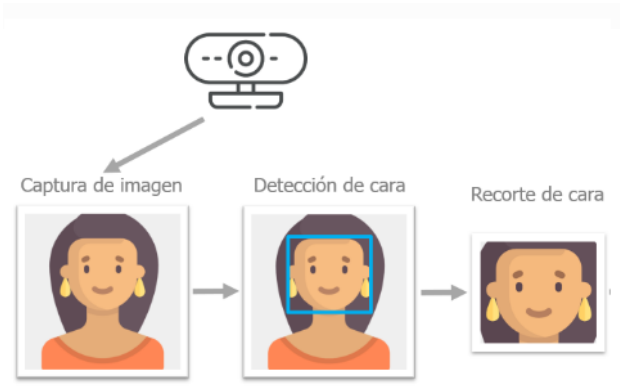

Fig. 1. Ejemplo de procesamiento de imagenes

B. Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales o CNN son un tipo de red neuronal que se utiliza principalmente para tareas de clasificacion de imagenes y reconocimiento de objetos, estan conformadas por capas: Input Layer, Convolutional Layer, Activation Layer, Pooling Layer, Fully Connected Layer, Output Layer.

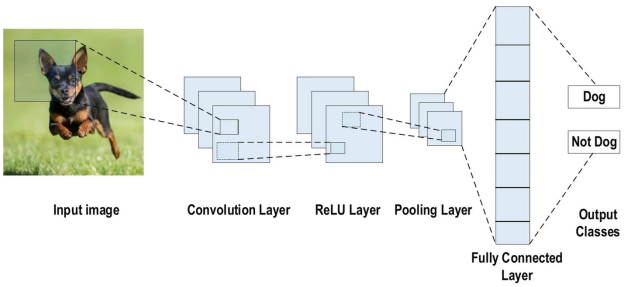

Fig. 2. Ejemplo de red neuronal convolucional

C. Keras

Keras es una biblioteca de codigo abierto para construir y entrenar modelos de apendrizaje profundo, esta libreria nos proporcionara una interfaz para trabajar con redes neuronales.

III. Arquitectura de las Redes Neuronales Convolucionales (CNN)

En esta seccion veremos a mayor profundidad cada capa de las CNN.

A. Input Layer

Esta capa recibe la imagen y le asigna una neurona a cada pixel de la imagen, donde cada pixel tiene un valor de entre 0-255. Por ejemplo en el caso en el que la imagen sea de tamaño n×n, la Input Layer tendra n^2 neuronas.

B. Convolutional Layer

En esta capa se realiza el proceso de convolucion donde la imagen que se ingresa es pasada por varias matrices pequeñas llamadas kernel (por ejemplo kernel de 3×3 o 5×5), cada casilla del kernel tiene un peso la cual va cambiando o ajustando, esto sucede durante el entrenamiento del modelo.
Este kernel se desplaza por la imagen "pixel por pixel", multiplicando los valores de los pixeles en la region local de la imagen por los valores del kernel y luego sumando estos productos para obtener un unico valor, este proceso lo realizara para todos los kernels, reduciendo el tamaño de la imagen.
El resultado de este proceso es un feature map (mapa de caracteristicas) que refleja la presencia de ciertos patrones en la imagen, como bordes u otros.

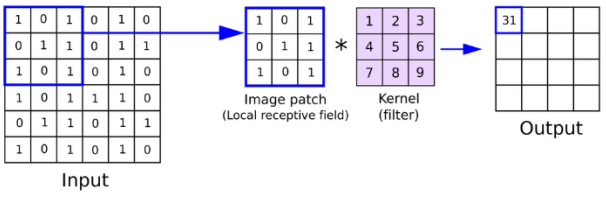

Fig. 3. Ejemplo de convolucion en la Convolutional Layer

Otros parametros importantes dentro de la Convolutional Layer son el stride (paso) y el padding (relleno):

- **stride:** El stride (paso) determinara como se movera el kernel en la imagen, por ejemplo si el stride es 1, el kernel se movera un pixel por cada paso y si el stride es 2, el kernel se movera 2 pixeles por cada paso. Esto hara que la imagen se reduzca de diferente forma dependiendo del stride (paso).
  El tamaño de las imagenes de salida de la Convolutional Layer osea osea el feature map segun una matriz de entrada n×n, kernel m×m y stride (paso) igual a s, es de:

$$((n-m)/k+1) \times ((n-m)/k+1)$$

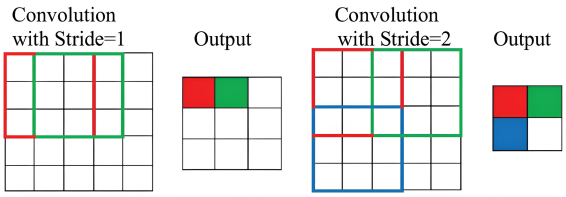

Fig. 4. Ejemplo con stride=1 y stride=2

- **padding:** El padding es el proceso de agregacion de capas adicionales de valores a las filas y columnas externas de la matriz de entrada. Existen 2 tipos principales de padding:
  - **Valid:** No agregara ninguna capa adicional de cero, osea sin padding.
  - **Same:** Se agregaran capas adicionales de ceros a las filas y columnas externas de la matriz de entrada, para que el feature map mantenga el mismo tamaño de la matriz de entrada.

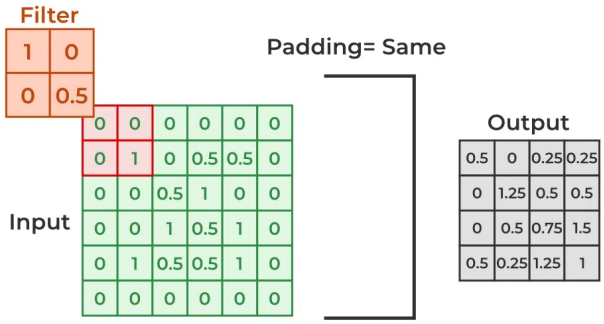

Fig. 5. Ejemplo de padding de tipo same

C. Activation Layer

En esta capa se aplicaran a los valores de entradas Activation Functions (funciones de activacion) con el proposito de introducir la no linealidad en la CNN, esto para que la CNN aprenda relaciones mas complejas.
Estos son los tipos de Activation Functions mas utilizados en la CNN:

- **Sigmoid:** Esta funcion tiene como salida valores entre 0 y 1, se representa de la siguiente manera:

$$f(x)_{\text{sigm}} = \frac{1}{1+e^{-x}}$$

- **Tanh:** Similar a la sigmoid function con la diferencia de que la salida esta entre -1 y 1, se representa de la siguiente manera:

$$f(x)_{\text{tanh}} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

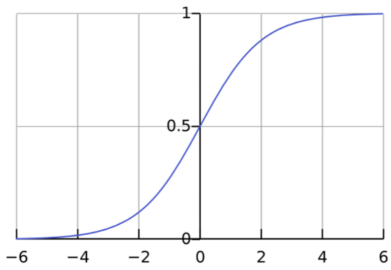

Fig. 6. Sigmoid Function

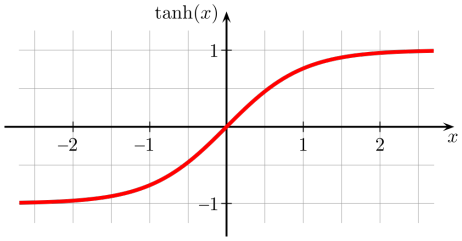

Fig. 7. Hyperbolic Tangent Function

- **ReLU:** Esta es la funcion mas utilizada en las CNN, la salida de esta funcion seran de solamente numero positivos, veamos 2 casos.
  En el caso de que el valor de entrada sea negativo, la salida sera 0.
  En el caso de que el valor de entrada sea positivo, la salida sera igual a la entrada, osea no cambiara.
  Esta funcion se representara de la siguiente manera:

$$f(x)_{\text{ReLU}} = \max(0, x)$$

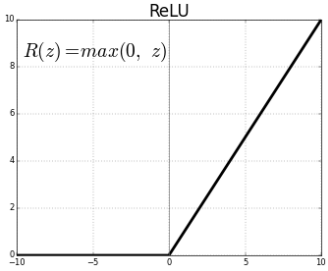

Fig. 8. Function ReLU

- **SoftMax:** Esta funcion convierte los valores de entrada en probabilidades, donde la suma de todas las probabilidades es 1, esta funcion se utilizara en la ultima capa de la Fully Connected Layer, se escribe de la siguiente manera:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

  Donde:
  - z_i es el valro de entrada para la cual estamos calculando la probabilidad.
  - e^{z_j} el exponente de z_i asegurara que los valores resultantes sean positivos.
  - $\sum_{j=1}^{n} e^{z_j}$ esta suma normaliza cada valor para que el resultado este entre 0 y 1, y que la suma total sea 1.

D. Pooling Layer

En esta capa, la entrada sera la salida de la Convolutional Layer, que viene siendo el feature map de cada kernel.
La idea general de esta capa es reducir el tamaño de los feature maps, de tal manera que resuman las caracteristicas que estan en los feature maps, asi reduciendo la cantidad de parametros y calculos en la CNN, haciendo el modelo mas robusto a las variaciones en la posicion de las caracteristicas en la imagen de entrada.Y como hace esto las Polling Layers?
Esto se hace pasando una matriz llamada pool (comunmente de tamaño 2×2) por las feature maps, el pool no repetira celdas por las que ya paso antes, este nuevo feature map lo llamaremos Pooled Feature Map, entre las formas de pooling existen 2 principales:

- **Max Pooling:** Se tomara el maximo valor dentro del pool.

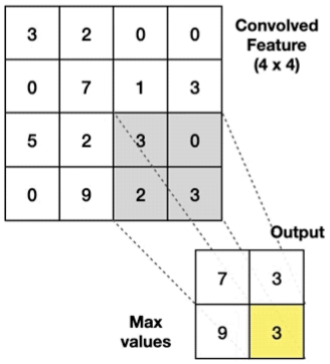

Fig. 9. Ejemplo de Max Pooling

- **Average Pooling:** Se tomara el promedio de los valores dentro del pool.

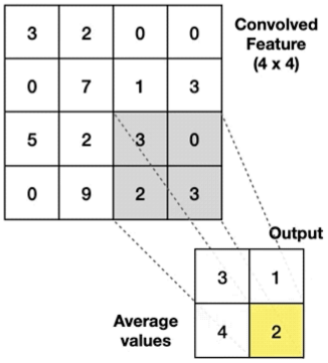

Fig. 10. Ejemplo de Average Pooling

E. Flattening

Este el ultimo proceso para pasar a las Fully Connected Layer, como el Pooled Feature Maps de la ultima Pooling Layer son matrices, necesitamos pasarlo a un vector para que sea compatible con la entrada de la Fully Connected Layer.

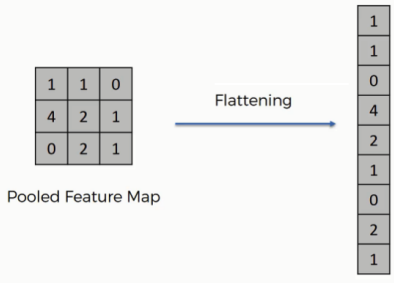

Fig. 11. Ejemplo de Flattening a un Pooled Feature Map

F. Fully Connected Layer

En esta capa se tomara las caracteristicas de la imagen, las procesara para hacer una prediccion final, ahora veamos como hace esto.
Dentro hay varias capas de neuronas llamadas Hidden Layers (Capas Ocultas), cada neurona esta conectada a todas las neuronas de la capa anterior, cada conexion con otra neurona tendra un peso y cada neurona tendra un sesgo, el valor de la siguiente neurona sera la suma ponderada de las neuronas y sus pesos, mas en detalla vendra siendo el producto de los valores de la neuronas anteriores por el peso de la conexion, agregandole mas el sesgo de la neurona, su representacion matematica seria de la siguiente forma:

$$y_i = \sum_{j=1}^{n} w_{ij} \cdot x_j + b_i$$

Donde:

- y_i es el valor de la neurona i de la capa actual.
- w_{ij} es el peso de la conexion entre la neurona j de la capa anterior y la neurona i de la capa actual.
- x_j es el valor de la neurona j en la capa anterior.
- b_i vendra siendo el sesgo de la neurona i de la capa actual.

Luego de calcular los valores de las neuronas de la siguiente Hidden Layer, estos valores se pasaran a traves de una Activation Function (Funcion de Activacion), como hemos visto anteriormente en la Activation Layer, la Activation Function introducira no linealidad, permitiendo a la CNN aprender representaciones complejas de los datos.
La ultima capa de la Fully Connected Layer sera donde se produzca la prediccion final de la CNN, en esta capa habra tantas neuronas como clases de imagenes que queremos clasificar o reconocer, donde a cada valor de la neurona se pasara una Activation Function en este caso sera SoftMax, la cual converitira estos valores en valores de probabilidades, donde la suma total de estos valores de propabilidad sera de 1, lo que nos facilitara el proceso de interpretacion de cada valor como la probabilidad de pertenecer a cada clase.
Durante el entrenamiento, estos pesos y sesgos de la Fully Connected Layer iran cambiado/ajustandose mediante el Gradient Descent (Descenso de Gradiente) y Backpropagation (retropropagacion) optimizando o minimizando la Loss Function (Funcion de Perdida), esta Loss Function calculara el error del entrenamiento, este error representara la diferencia entre el resultado real y el previsto por la CNN.
Todo esto con el fin de que nuestro modelo pueda predicir exitosamente que ”cosa” es la imagen que se esta mostrando.

IV. Loss Function

A. Mean Squared Error

La Mean Squared Error (Error Cuadratico Medio) es una forma de calcular el error del entramiento en cada epoca, esta se escribe de la siguiente manera:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Donde:

- n es el numero de muestras en el conjunto de datos.
- y_i es la etiqueta verdadera de la muestra i.
- $\hat{y}_i$ es la salida o valor predicha por el modelo,

B. Mean Absolute Error

La Mean Absolute Error (Error Absoluto Medio) es otra forma de calcular el error de entramiento, esta se escribira de esta forma:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- n es el numero de muestras en el conjunto de datos.
- y_i es la etiqueta verdadera de la muestra i.
- $\hat{y}_i$ es la salida o valor predicha por el modelo,

C. Binary Cross Entropy

Es una Loss Function utilizada para problemas de clasificacion binaria, donde solo hay 2 opciones de clases, esta funcion se escribe de la siguiente manera:

$$-\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Donde:

- n es el numero de muestras en el conjunto de datos.
- y_i es la etiqueta verdadera de la muestra i (0 o 1).
- $\hat{y}_i$ es la salida o valor predicha por el modelo,

D. Categorical Cross Entropy

Es una Loss Function utilizada principalmente en problemas de clasificacion multiclase, donde a contrario a Binary Cross Entropy se tiene mas de dos categorias posibles, esta funcion se escribe de la siguiente manera:

$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(p_{ij})$$

Donde:

- n es el numero de muestras en el conjunto de datos.
- m es el numero de clases.
- y_{ij} es un valor binario que indica si la muestra i pertenece a la clase j (1 si es cierto o 0 si no).
- p_{ij} es la probabilidad que da el modelo de que la muestra i pertenece a la clase j.

V. Gradient Descent

El Gradient Descent (Descenso de Gradiente) es un algoritmo iterativo que minimiza el error de entramiento o Loss Function, este algoritmo actualizara los parametros de la CNN como los pesos y sesgos, esto para cada epoca de entrenamiento.
Para esto calcularemos la gradiente de la Loss Function aplicando una derivada de primer orden con respecto a un parametro de la CNN (por ejemplo el peso y el sesgo).
Estos pesos y sesgos se actualizaran en direccion inversa del gradiente, el proceso de actualizacion de parametros se dara en la Backpropagation de la CNN, en la que la gradiente en cada neurona se propaga hacia atras a todas las neuronas de la Hidder Layer anterior, este algoritmo iterativo se representa de la siguiente manera para los pesos:

$$w_{ij}^{t} = w_{ij}^{t-1} - \Delta w_{ij}^{t}$$

$$\Delta w_{ij}^{t} = \eta \times \frac{\partial E}{\partial w_{ij}}$$

Donde:

- w_{ij} es el peso de la conexion entre la neurona j de la capa anterior y la neurona i de la capa actual.
- w_{ij}^{t} es el peso final en la epoca de entrenamiento actual t.
- w_{ij}^{t-1} es el peso en la epoca de entrenamiento anterior $(t-1)$.
- η es la tasa de aprendizaje que se define como el tamaño del paso de la actualizacion del parametro en este caso el peso.
- E es la Loss Function.

En el caso de los sesgos es similar:

$$b_{i}^{t} = b_{i}^{t-1} - \Delta b_{i}^{t}$$

$$\Delta b_{i}^{t} = \eta \times \frac{\partial E}{\partial b_{i}}$$

Donde:

- b_i es el sesgo de la neurona i de la capa actual.
- b_i^t es el sesgo final en la epoca de entrenamiento actual t.
- b_i^{t-1} es el sesgo en la epoca de entrenamiento anterior $(t-1)$.
- η es la tasa de aprendizaje que se define como el tamaño del paso de la actualizacion del parametro en este caso el sesgo.
- E es la Loss Function.

VI. Backpropagation

La Backpropagation (Retropropagacion) es un algoritmo que se utiliza para entrenar una CNN, actualiza los pesos y sesgos para minimizar la Loss Function para mejorar el rendimiento de nuestra CNN.
Lo podemos separar en 3 pasos, el forward pass (pase hacia adelante), el loss calculation (calculo de perdidad o error)y el backward pass (pase hacia atras).

A. Forward Pass

Durante este paso los datos de entrada, en este caso imagenes, pasaran atraves de la CNN para determinar la salida, como ya vimos esta imagen de entrada pasara por las Convolutional Layers, Ac- tivation Layers, Pooling Layers y Fully Connected Layers.

B. Loss Calculation

Despues de obtener la salida de la CNN, se obtiene una Loss Function la cual calculara el error de entranamiento de esa epoca, podemos utilizar la Loss Function que mas nos convengan, vemos que en el caso de la clasificacion de imagenes de multiclase, viene bien utilizar la Categorical Cross Entropy.

C. Backward Pass

Para hacer el Backward Pass, calcularemos el gradiente de la Loss Function con respecto a cada peso en la CNN, veremos que comenzaresmo de adelante hacia atras y aplicaremos la regla de la cadena.

- **Gradiente con Respecto a la Salida:** Primero se calculara el gradiente de la loss function con respecto a la salida de la capa final ($\hat{y}$):

$$\frac{\partial E}{\partial \hat{y}}$$

- **Gradiente a traves de la activation function:** Luego este gradiente se propagara a traves de la activation function, usaremos la regla de la cadena a traves de la activation function.

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial x}$$

  Donde x es el valor de entrada de la activation function, que es el valor que llega a la neurona antes de aplicarle la activation function.
- **Gradiente con respecto a los kernels en convolutional layer:** Continuaremos aplicando la regla de la cadena para encontrar la gradiente con respecto a los kernels.

$$\frac{\partial E}{\partial K} = \frac{\partial E}{\partial \hat{y}} * \text{rot180}(I)$$

  Donde:
  - **K** es el kernel.
  - **I** es la entrada a la convolutional layer, es el valor que recibe la convolutional layer.
  - **rot180(I)** indica que la entrada I se rota 180 grados antes de hacer la convolucion inversa. Esto sera necesario para alinear el calculo del gradiente de la misma forma en que se aplico la convolucion durante la propagacion hacia adelante.

  Al aplicar este gradiente se ajustaran los valores de cada kernel en la convolutional layer, lo que va a permitir que los kernels se adapten para identificar mejor las caracteristicas de la imagen que se ingresa.

Esto tres procesos se realizara varias veces en bucle hasta tener bien ajustado los parametros de la CNN para identificar imagenes.

REFERENCES

[1] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). Link

[2] Zhang, Z. (2016). Derivation of Backpropagation in Convolutional Neural Network (CNN). University of Tennessee, Knoxville, TN. Link

[3] Deepgram. (2024, 16 de junio). Loss Function. Recuperado de Link

[4] GeeksforGeeks. (2024, 8 de septiembre). Backpropagation in Convolutional Neural Networks. Recuperado de Link

[5] Purwono, Purwono and Ma'arif, Alfian and Rahmaniar, Wahyu and Imam, Haris and Fathurrahman, Haris Imam Karim and Frisky, Aufaclav and Haq, Qazi Mazhar Ul. (2023). Understanding of Convolutional Neural Network (CNN): A Review. Link

[6] Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018). Link