Descripción del Algoritmo Usado Para el SACP

J. C. Barrera Guevara[1], D. A. Machado Tovar[2], D. F. Baquero Cerquera[3], J. Gregorio Delgado[4], J. S. Forero Agudelo[5], C. A. Pérez Ochoa[6], M. F. Santofimio Romero[7]

Facultad de Ciencias Básicas e Ingeniería, Universidad de los Llanos

Villavicencio, Colombia

jc.bguevara@unillanos.edu.co[1]

damachado@unillanos.edu.co[2]

dfbaquero@unillanos.edu.co[3]

jg.delgado@unillanos.edu.co[4]

jsforero.agudelo@unillanos.edu.co[5]

caperez.ochoa@unillanos.edu.co[6]

mfsantofimio@unillanos.edu.co[7]

Convolución

La convolución es una operación matemática fundamental en áreas como el procesamiento de señales, el análisis de imágenes, la física y el aprendizaje automático.

$$(f*g)(t) = \int_{-\infty}^{\infty} f(\tau)\cdot g(t-\tau)d\tau$$

Ecuación 1. Definición matemática de la convolución.

Para el dominio discreto, como en el análisis de imágenes, la convolución se expresa como se muestra en la ecuación 2.

$$(I*K)(i,j) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} I(i-m, j-n)\cdot K(m,n)$$

Ecuación 2. Definición matemática de la convolución discreta.

- I = Imagen de entrada (matriz 2D).
- K = Kernel o filtro (matriz pequeña, típicamente 3×3 o 5×5).
- (i,j) = Posición en el mapa de características de salida.

En esencia, lo que hace esta operación matemática es usar el kernel K, el cual funciona como un filtro, y "barrer" toda la imagen, la cual es representada como I para el dominio discreto. Esto quiere decir que dependiendo como esté configurado el kernel vamos a obtener uno u otro resultado.

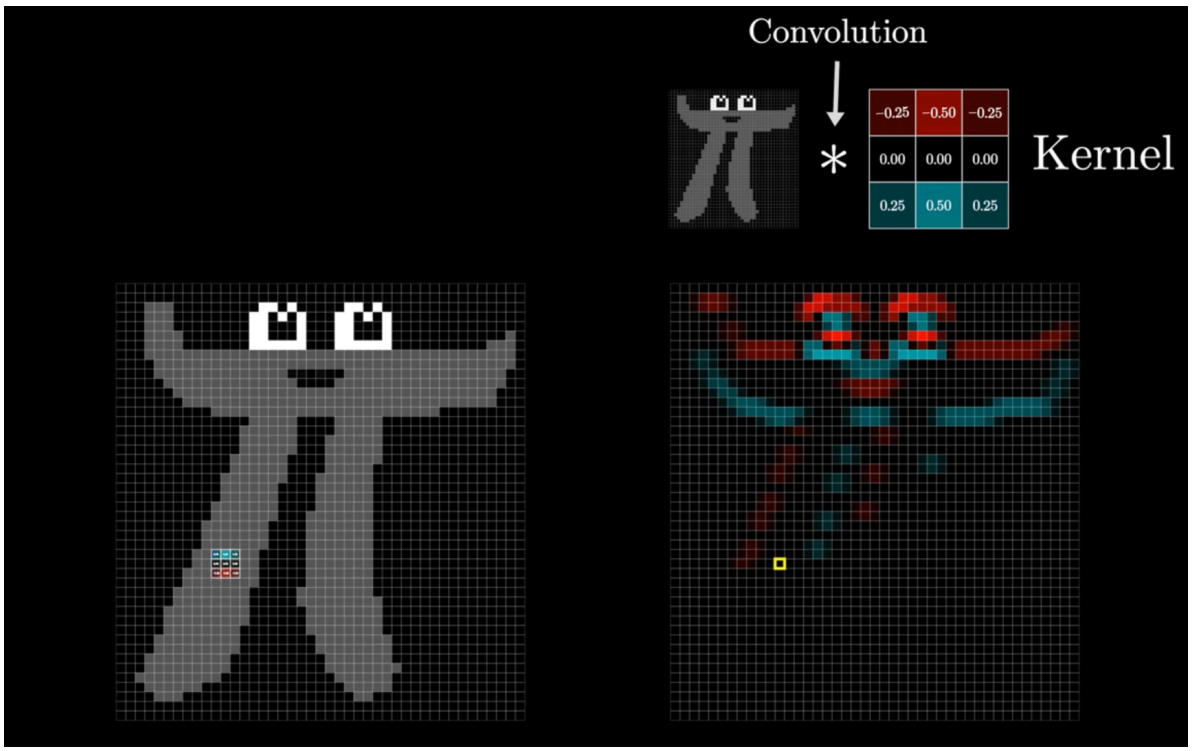

Figura 1. Ejemplo de convolución donde se procesa una imagen con un kernel específico. Reproducido de 3Blue1Brown. (2017, 5 octubre). But what is a neural network?

Redes Neuronales

Una red neuronal es un sistema de aprendizaje automático inspirado en el funcionamiento del cerebro humano, donde pequeñas unidades llamadas neuronas artificiales trabajan en conjunto para procesar información.

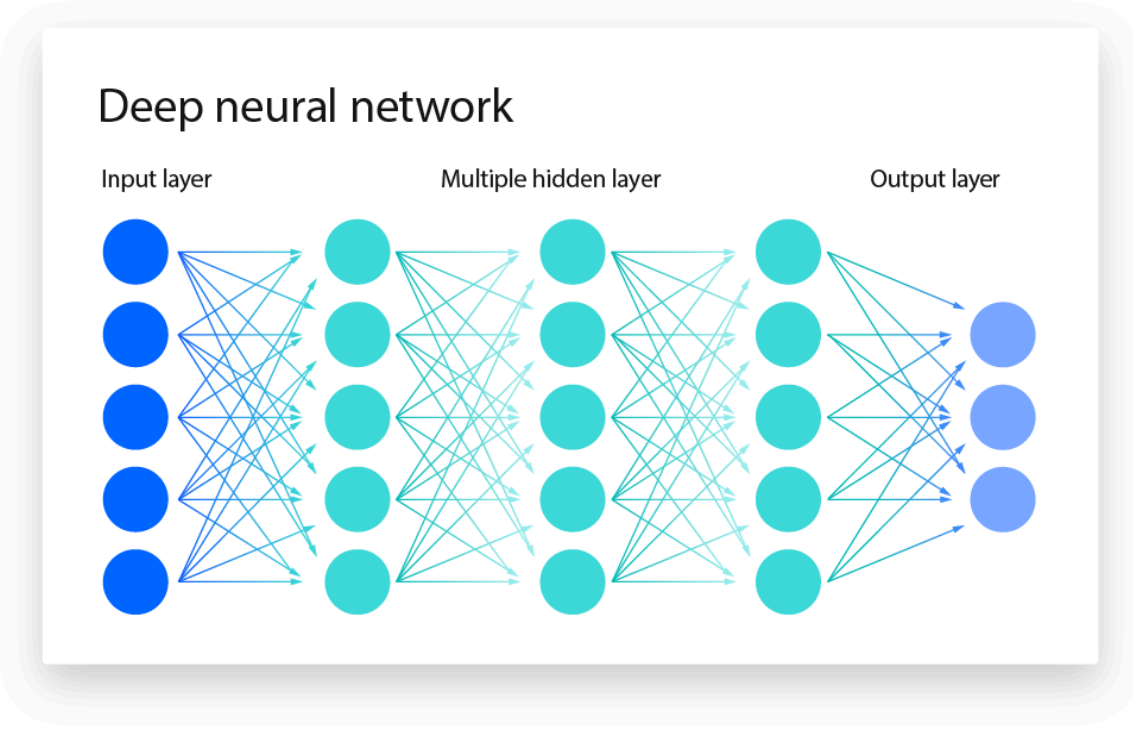

Figura 2. Diagrama de una red neuronal. Reproducido de IBM. (s.f.). Redes neuronales.

Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN o ConvNets) son un tipo especial de red neuronal diseñada para procesar datos con estructura espacial, como imágenes, de manera eficiente y jerárquica. Su funcionamiento se basa en tres ideas clave: convoluciones locales, sharing de pesos y aprendizaje jerárquico de características.

● Convoluciones (detectores de patrones locales):

La CNN usa pequeños filtros (kernels), como lupas de 3x3 píxeles, que se deslizan sobre la imagen. Cada filtro está especializado en detectar patrones básicos: bordes horizontales, manchas, cambios de color, etc. Estos filtros no son programados a mano, sino que se aprenden durante el entrenamiento. Por ejemplo, un filtro podría activarse al encontrar los ojos de un pescado.

● Mapas de características y no-linealidad (ReLU):

Cada filtro genera un "mapa de características" que resalta dónde aparece su patrón asignado. Luego, una función de activación (como ReLU) elimina valores negativos, dejando solo las características relevantes. Esto introduce no-linealidad, permitiendo a la red aprender relaciones complejas.

● Pooling (reducción inteligente):

Para evitar que la red se pierda en detalles irrelevantes, el "pooling" (como max-pooling) reduce el tamaño de los mapas de características conservando sólo la información más importante. Por ejemplo, si en una región de 2x2 píxeles hay un borde fuerte, pooling guarda ese dato y descarta el ruido.

● Capas profundas (jerarquía de abstracción):

Las primeras capas detectan patrones simples (bordes, texturas), las intermedias combinan esos patrones en formas (orejas, ojos), y las últimas capas ensamblan todo para reconocer objetos completos (un rostro de gato). Esto se logra apilando múltiples capas convolucionales, cada vez con más filtros especializados.

● Capa fully-connected (decisión final):

Al final, la CNN "aplana" los mapas de características y usa neuronas tradicionales para decidir, por ejemplo, "90% de probabilidad de que sea un gato".

MobileNetV2

MobileNetV2 es una arquitectura revolucionaria de red neuronal convolucional que prioriza la eficiencia computacional sin sacrificar precisión, logrando un equilibrio perfecto entre rendimiento y versatilidad. Su diseño inteligente se basa en principios innovadores que redefinen cómo las redes procesan información visual.

Bloques de Residuales Invertidos (Inverted Residuals)

MobileNetV2 reinventa el flujo de datos mediante una estructura única:

Expansión inicial

Primero proyecta los canales de entrada a un espacio de mayor dimensionalidad (usando convoluciones 1×1), creando una "representación intermedia rica". Por ejemplo, si entra con 64 canales, puede expandir a 384.

Convolución profunda

En este espacio ampliado, aplica filtros 3×3 por canal separadamente (depthwise convolution), extrayendo patrones espaciales de manera eficiente.

Compresión con linealidad

Finalmente, reduce los canales otra vez con una convolución 1×1, pero omite la no-linealidad (ReLU) en este paso. Esto evita que se pierda información crítica al comprimir, un fenómeno llamado "colapso de dimensionalidad".

Arquitectura

- **Capa inicial**

Una convolución estándar 3×3 extrae features básicos (bordes, texturas).

- **Secuencia de bloques invertidos (15 en total)**

Cada bloque opera a diferentes resoluciones (desde 224×224 hasta 7×7 píxeles). Usa strides (pasos) de 2 para reducir dimensiones cuando corresponda.

- **Capa final**

Convolución 1×1 transforma todos los features en un vector para clasificación.

Pseudocódigo del algoritmo diseñado

```
Unset
INICIO
1. PREPROCESAMIENTO DE IMÁGENES (módulo: preprocess.py)
    Para cada imagen:
        - Remover fondo con Rembg (ONNX)
        - Detectar cuerpo (mayor contorno en la máscara)
        - Detectar ojos usando clasificador Haar (OpenCV)
        - Dibujar rectángulos en imagen (ojos, cuerpo)
        - Guardar imagen segmentada en carpeta temporal
        - Redimensionar a 224x224 píxeles
        - Normalizar los valores de los píxeles [0-1]
        - Expandir dimensión para formato de lote (batch)

2. ENTRENAMIENTO DEL MODELO (módulo: train.py)
    - Leer Excel con nombres de imágenes y calificaciones (ojos/piel)
    - Convertir rutas relativas en rutas absolutas
```

```
    - Normalizar calificaciones al rango [0, 1]
    - Dividir datos en conjuntos de entrenamiento y validación
    - Crear generadores de datos por lote:
        • Leer imagen, convertir a array y normalizar
        • Asociar con etiquetas correspondientes
    - Cargar MobileNetV2 (sin la capa final, con pesos ImageNet)
    - Congelar sus pesos (no entrenables)
    - Añadir:
        • GlobalAveragePooling
        • Capa densa oculta
        • Capa densa con 2 salidas (ojos, piel), activación sigmoide
    - Compilar modelo con:
        • Pérdida MSE
        • Optimizador Adam
        • Métrica MAE
    - Entrenar por 10 épocas
    - Guardar el modelo en disco (modelo_entrenado.h5)

3. ANÁLISIS / INFERENCIA (módulo: analyze.py)
    Para una imagen o conjunto de imágenes:
        - Preprocesar imagen como en paso 1
        - Cargar modelo entrenado
        - Realizar predicción
        - Convertir salida (valores entre 0-1) a escala de 0 a 5
        - Retornar resultados:
            • Calificación ojos (0–5)
            • Calificación piel (0–5)
            • Ruta de imagen procesada (con anotaciones)

FIN
```

Bibliografía

Howard, A. G. et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861.

IBM Think Topics. (2023). Redes Neuronales: Conceptos y Aplicaciones. https://www.ibm.com/es-es/think/topics/neural-networks

Computerphile. (2020). How Convolutional Neural Networks Work [Video]. YouTube. https://www.youtube.com/watch?v=KuXjwB4LzSA

TensorFlow Documentation. (2023). MobileNetV2. https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV2

PyTorch Tutorials. (2023). Depthwise Separable Convolutions. https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html