Modelo_Resunet

March 30, 2024

1 Arquitectura- ResUnit

Nombre y Apellidos: Javier Ochoa De La Cruz Google https://colab.research.google.com/drive/1Ke16s2jhFj8GBF_XYgVVYp1EqaTo4g1D?usp=sharing

[ ]:
```
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation,␣
 ↪MaxPool2D, UpSampling2D, Concatenate, Input
from tensorflow.keras.models import Model

def Normalizacion_Relu(inputs):
     # Aplica Normalizacion & Funcion ReLU
     x = BatchNormalization()(inputs)
     x = Activation("relu")(x)
     return x


# Se define una funcion para identificar todo proceso de lo nuevo de la␣
 ↪arquitectura ResUnet que maneja bloques residales
# que maneja comunicacion entre capas para mejorar el flujo de gradientes a␣
 ↪través de la red durante el entrenamiento.

def bloque_residual (inputs, num_filters, strides=1):

     # Parte 1=> Capa Convolucional
     capa1 = Normalizacion_Relu(inputs)
     # toma valor de strides que se envia parametro a la funcion llamada
     capa1 = Conv2D(num_filters, kernel_size=(3, 3), padding="same",␣
 ↪strides=strides)(capa1)
     capa1 = Normalizacion_Relu(capa1)
     # toma valro stride = 1
     capa1 = Conv2D(num_filters, kernel_size=(3, 3), padding="same",␣
 ↪strides=1)(capa1)

     # Conexiones directas o Mapeo de Identidad lo nuevo por Resunet

     capa2 = Conv2D(num_filters, kernel_size=(1, 1), padding="same",␣
 ↪strides=strides)(inputs)
```

```
     salida_residual = capa1 + capa2
     return salida_residual

def bloque_decoder(inputs, skip_features, num_filters):

    x=UpSampling2D((2,2))(inputs)
    x=Concatenate()([x, skip_features])
    x=bloque_residual(x, num_filters, strides=1)
    return x

def arquitectura_resunet(input_shape):

    # arquitectura Resinet

    inputs = Input(input_shape)

    # **Encoder etapa1**

    etapa1 = Conv2D(filters=64, kernel_size=(3, 3), padding="same",␣
 ↪strides=1)(inputs)
    etapa1 = Normalizacion_Relu(etapa1)
    etapa1 = Conv2D(filters=64, kernel_size=(3, 3), padding="same",␣
 ↪strides=1)(etapa1)

    # Aplica la estructura Addition suma convolucion kernel (1,1) de input␣
 ↪original + salida de econder1
    # tener en cuenta que usa 1x1 para igualar las dimensiones de los canales␣
 ↪entre la entrada y la salida de un bloque residual
    # reducir la profundidad (número de canales) de los mapas de␣
 ↪características menos cant de cálculos de opeaciones, mejorando la␣
 ↪eficiencia del modelo
    # sin perder mucha información relevante

    salida = Conv2D(filters=64, kernel_size=(1, 1), padding="same")(inputs)
    print("num ope salida",salida)
    # aplica Addition segun estrcutura del articulo

    salida1 = etapa1 + salida

    # Encoder etapa 2 y 3

    salida2 =bloque_residual(salida1, 128, strides=2)
    salida3 =bloque_residual(salida2, 256, strides=2)

    # Bridge

    bridge =bloque_residual(salida3, 512, strides=2)
```

```
    # Decoder 1, 2, 3 de abajo hacia arriba

    salida_dec = bloque_decoder(bridge, salida3, 256)
    salida_dec = bloque_decoder(salida_dec, salida2, 128)
    salida_dec = bloque_decoder(salida_dec, salida1, 64)

    # Classificador en la salida aplica convolucion y sigmoidal a la salida del␣
 ↪decodificar

    outputs = Conv2D(1, 1, padding="same", activation="sigmoid")(salida_dec)

    # Creamos nuestra arquitectura de red: como se necesita definir como fluyen␣
 ↪los datos a través del modelo
    # donde tenemos operaciones que no son secuenciales donde algunas capas se␣
 ↪suma con sus entradas ya no
    # usaremos secuencial sino uno para modleos complejos importaremos API␣
 ↪funcional (Model)

    model = Model(inputs, outputs, name="Arquitectura_Resinet")
    return model

#tomaremos de entrada segun el modelo  224 pixels de altura , 224 de ancho y  3␣
 ↪de profundidad RGB, ademas considerar
# que este modelo La elección de 224x224x3 como forma de entrada permite␣
 ↪reutilizar este modelo a futuro en investigacion
# preentrenado en otras tareas mediante técnicas de transferencia de␣
 ↪aprendizaje, modificando algunas capas de salida.
# llamda incial al modelo

shape=(224, 224, 3)

# llamamos al modelo
model_arquitectura_resunet = arquitectura_resunet(shape)

# mostramos detalle parametros del modelo
model_arquitectura_resunet.summary()
```