PRÁCTICA 1 - VPC

Wide ResNet on Cifar10

Cristian Villarroya Sánchez

01-06-2021

Índice

1. Descripción de la tarea

Se ha realizado la implementación de una Wide Resnet sobre el dataset Cifar10.

La implementación de la Wide ResNet se ha realizado con la ayuda del ejemplo disponible en el GitHub [1]. Con esta implementación, se permite crear cualquier tipo de Wide Resnet, en función de los parámetros que se establezcan en la función para crear la misma.

2. Modelo

En este caso concreto, se ha realizado una implementación de una Wide ResNet 16 con profundidad 8.

La Wide Resnet se divide en 5 partes:

- Convolución inicial:
  - Convolucion2d 3x3 de profundidad 16
  - Batch Normalization
  - Activacion Relu
- Convolución expand:
  - Es una convolución y una capa que se salta la convolución (residual)
  - Conv 2d 3x3 de la profundidad que se establezca
  - Batch Normalization
  - Relu
  - Conv 2d 1x1 para la residual (no modifica)
  - La salida es la suma de la convolución y la residual
- Convolución 1:
  - Batch Normalization
  - Activacion Relu
  - Convolucion2d 3x3 de profundidad 16
  - Dropout si se especifica
  - Batch Normalization
  - Activacion Relu
  - Convolucion2d 3x3 de profundidad 16
  - La salida es la suma de las dos convoluciones

- Convlución 2:
  - Batch Normalization
  - Activacion Relu
  - Convolucion2d 3x3 de profundidad 32
  - Dropout si se especifica
  - Batch Normalization
  - Activacion Relu
  - Convolucion2d 3x3 de profundidad 32
  - La salida es la suma de las dos convoluciones

- Convolución 3:
    - Batch Normalization
    - Activacion Relu
    - Convolucion2d 3x3 de profundidad 64
    - Dropout si se especifica
    - Batch Normalization
    - Activacion Relu
    - Convolucion2d 3x3 de profundidad 64
    - La salida es la suma de las dos convoluciones

El parámetro N es el que controla la cantidad de bloques convolucionales (sin contar el bloque inicial y el bloque expand) que se van a implementar. Al final de cada uno de estos bloques se aplica también un Batch Normalization y una activación Relu.

Después, al final de la última convolución (convolución 3) se aplica Batch Normalization, activación Relu y a continuación un Average Pooling y Flatten, para convertir el tensor a un vector, que, a su vez, se conecta con una capa densa de tamaño 10, que es el número de clases diferentes en Cifar10.

Con esto, se ha creado el modelo de la siguiente manera:

```
model = create_wide_residual_network(x_train.shape[1:], nb_classes=num_classes, N=2, k=8, dropout=0.00)
```

Ilustración 1. Creacion de la Wide ResNet

La configuración del modelo es la siguiente:

- Data Augmentation
    - width_shift_range=0.17
    - height_shift_range=0.17
    - horizontal_flip=True
    - zoom_range=0.17
- Optmizador Adam
- Learning rate 0.01
- Model Checkpoint para guardar la mejor ejecución
- 60 epocas de entrenamiento
- Droupout 0.1
- Wide Resnet 16, profundidad 8

Tras 60 épocas de entrenamiento, el modelo alcanzo un 90.13% de precisión. No es de los mejores resultados del estado del arte con el dataset Cifar10 pero seguramente con una estructura más profunda se obtendrían mejores resultados, o incluso dejándolo unas épocas más.

```
Epoch 00059: val_acc improved from 0.90020 to 0.90130, saving model to weights.best.hdf5
Epoch 60/200
500/500 [==============================] - 180s 360ms/step - loss: 0.1104 - acc: 0.9613 - val_loss: 0.4625 - val_acc: 0.8889

Epoch 00060: val_acc did not improve from 0.90130
Epoch 61/200
105/500 [=====>........................] - ETA: 2:13 - loss: 0.1013 - acc: 0.9620
```

3. Conclusiones

Se ha implementado una wide resnet de tal manera que permita diferentes implementaciones (wide resnet16, wide resnet 28, wide resnet 40, etc) y se ha realizado una ejecución sobre el conjunto de datos Cifar10 obteniendo unos resultados decentes, aunque por debajo del estado del arte.

4. Bibliografía

[1]. Somshubra Majumdar, “Wide Residual Networks” https://github.com/titu1994/Wide-Residual-Networks , 2016