Visión por Computador

Práctica 2: CNN

Antonio Coín Castro

Curso 2019-20

Estructura del código

Se han desarrollado una serie de funciones genéricas de representación de imágenes, extráidas en su mayoría de la práctica 0 (aunque con ligeras modificaciones). Hay una serie de consideraciones a tener en cuenta:

- En general, todas las imágenes se convierten a números reales en el momento en que se leen, y solo se normalizan a $[0, 1]$ cuando se vayan a pintar.
- Las imágenes se pintan usando la librería `matplotlib`, empleando la técnica de *subplots* cuando sea necesario mostrar más de una imagen en la misma ventana.
- Tras mostrar una ventana de imagen, el programa quedará a la espera de una pulsación de tecla para continuar con su ejecución (incluso si se cierra manualmente la ventana del *plot*).
- Hay una serie de parámetros globales (editables) al inicio del programa para modificar ciertos comportamientos de algunas funciones.
- Se trabaja salvo excepciones con una única imagen de ejemplo en todo el programa, la cual debe estar en la ruta relativa `imagenes/`.
- Todas las funciones están comentadas y se explica el significado de los parámetros.

El programa desarrollado va ejecutando desde una función `main` los distintos apartados de la práctica uno por uno, llamando en cada caso a las funciones que sean necesarias para mostrar ejemplos de funcionamiento. Los parámetros de ejecución se fijan en estas funciones, llamadas `bonusX` ó `exXY`, donde `X` es el número de ejercicio e `Y` es el apartado.

Nota: Es posible que al ejecutar el programa aparezca el siguiente error: *Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).* Esto se debe a errores de redondeo en los cálculos, que hacen que las imágenes se salgan del rango $[0, 1]$ por valores despreciables del orden de 10^{18}.

- categorical_crossentropy is another term for multi-class log loss (http://wiki.fast.ai/index.php/Log_Loss

def logloss(true_label, predicted, eps=1e-15): p = np.clip(predicted, eps, 1 - eps) if true_label == 1: return -log(p) else: return -log(1 - p)

- Cortamos en 25 épocas porque se observa que el accuracy de validación comienza a bajar a partir de este número:

Ejemplo de pirámide Gaussiana de 4 niveles.

- Batchnormalization already includes the addition of the bias term. So there is no need (and it makes no sense) to add another bias term in the convolution layer. Simply speaking BatchNorm shifts the activation by their mean values. Hence, any constant will be canceled out. bn = gamma * normalized(x) + bias
- comentar los archivos temporales que se crean
- whithening no funciona bien
- Temp = True o False
- Explicar la función de compare()
- activation antes o después de BatchNorm
- explicar dropout

Basenet: 26 épocas, mejores pesos en la época 16.

———- BASENET MODEL EVALUATION ———- Test loss: 2.017395290565491 Test accuracy: 0.4452

Improved: 84 épocas, mejores pesos en la época 74

———- IMPROVED_BASENET MODEL EVALUATION ———- Test loss: 0.9160149734497071 Test accuracy: 0.7256

- APARTADO 3

CARACTERÍSTICAS:

———- CONV_MODEL EVALUATION ———- Test loss: 3.1692161699195056 Test accuracy: 0.38806462246633544

———- FC_MODEL EVALUATION ———- Test loss: 2.418268212735987 Test accuracy: 0.45268710853241473

FINE TUNING:

Red normal solo cambiando softmax y entrenando esa capa 10 épocas, sin entrenar el resto (fotos: _softmax).

———- FINE_TUNING EVALUATION ———- Test loss: 2.897768755805652 Test accuracy: 0.3003626772172766

Red añadiendo una capa fully connected con 1024 y un dropout(0.5) y entrenando todo 15 épocas.

———- FINE_TUNING EVALUATION ———- Test loss: 2.397986556532407 Test accuracy: 0.4497197494230135