VPC - Arquitectura YoloV3

Alejandro Granados Bañuls <algrabau@inf.upv.es>

Introducción

En este trabajo se va a desarrollar la arquitectura *YoloV3* en *PyTorch*. Para comprobar que se ha desarrollado correctamente tanto la topología de la red cómo la función de pérdida se va a usar el dataset *PASCAL VOC 2012* así cómo Tensorboard.

Yolo (You only look once) son un tipo de redes neuronales de visión por computador que sirven para detectar objetos en objetos en imágenes. Hay varios tipos de redes Yolo, en este trabajo se va a implementar la red *YoloV3* especificada en el paper YOLOv3: An Incremental Improvement. La implementación se va a basar en conseguir desarrollar correctamente la arquitectura así cómo la función de pérdida, el ajuste para obtener buenos resultados en *training* quedará pendiente cómo trabajo futuro.

YoloV3 consta de dos partes, una es la *Darknet-53* (qué podemos denominar como la *backbone* de la red), cuya topología puede verse en la Figura 1, y una parte de detección a varias escalas. La topología completa se puede ver en la Figura 2. Las escalas se pueden definir cómo diferentes divisiones en celdas de una imágen. Las escalas en *YoloV3* son 13x13, 26x26 y 52x52. Para que una imagen que se proprciona a una red pueda tener esas escalas deberá tener cómo tamaño 416x416.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figura 1: Topología Darknet-53

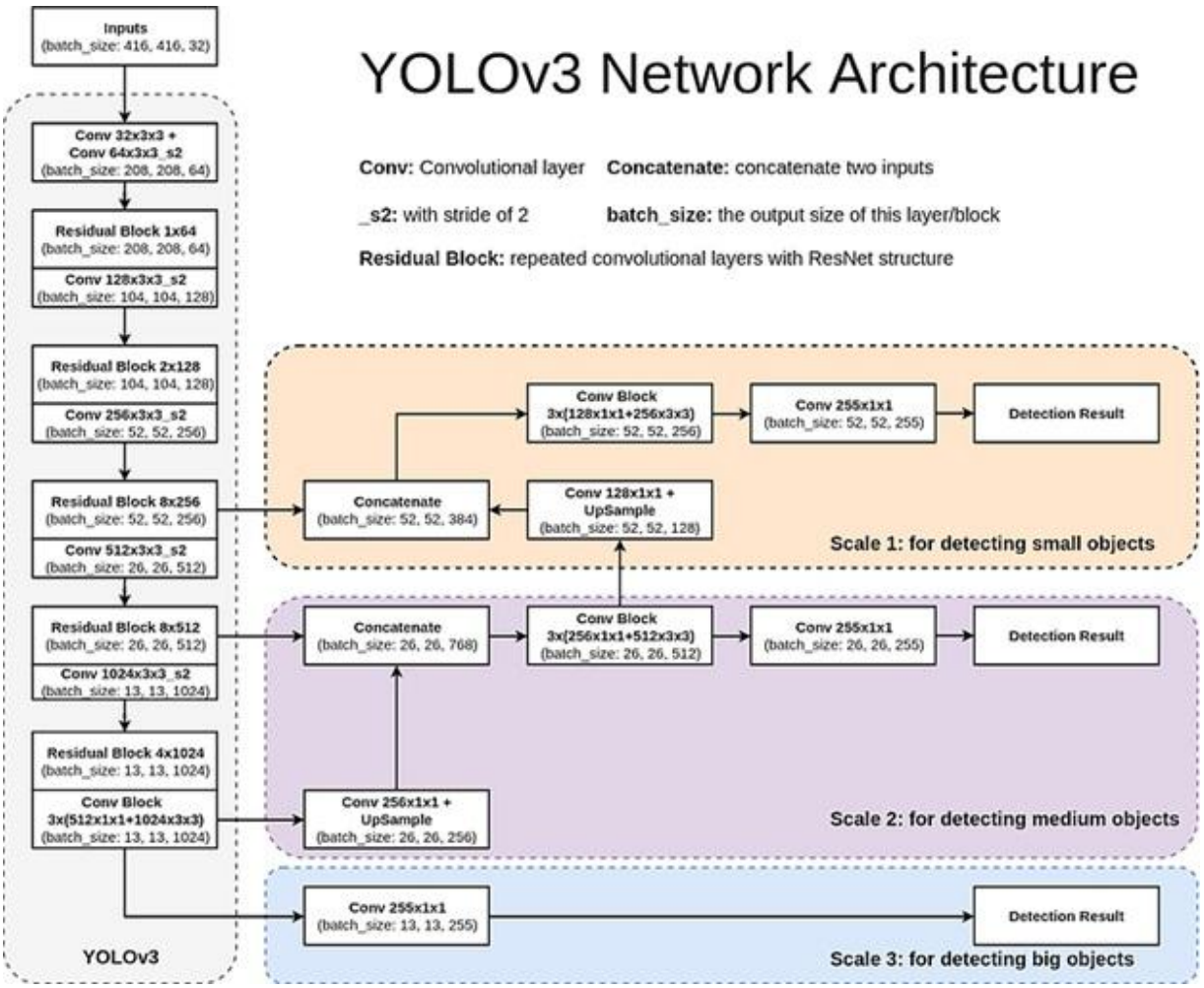

Figura 2: Topología YoloV3

YoloV3 usa *anchors* para las predicciones (cómo YoloV2/Yolo9000). Más específicamente usa tres anchors predefinidos (obtenidos a partir del análsis de las *bounding boxes* del dataset usando *k-means*) por escala para cada celda, por lo que en total se tienen 9 anchors (3 anchors x 3 escalas). Un ejemplo de esto se puede ver en la FIgura 3. Para predecir las coordenadas reales de la imágen se usa la fórmula representada en la Figura 4, donde *sigma* significa *Sigmoide*, t_x, t_y, t_w y t_h representan 4 de las salidas de la red.

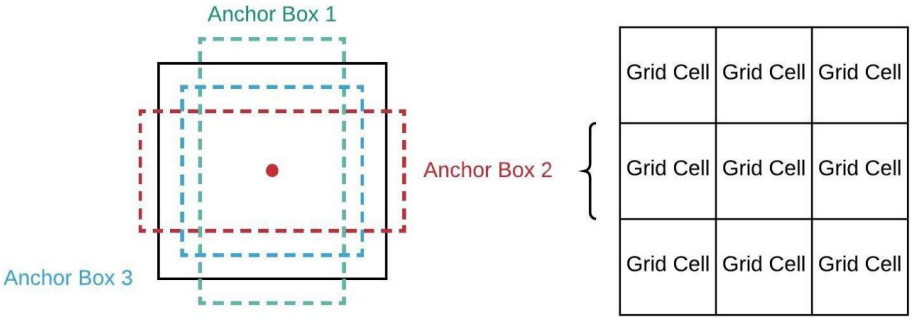

Figura 3: Ejemplo de los anchors de una celda

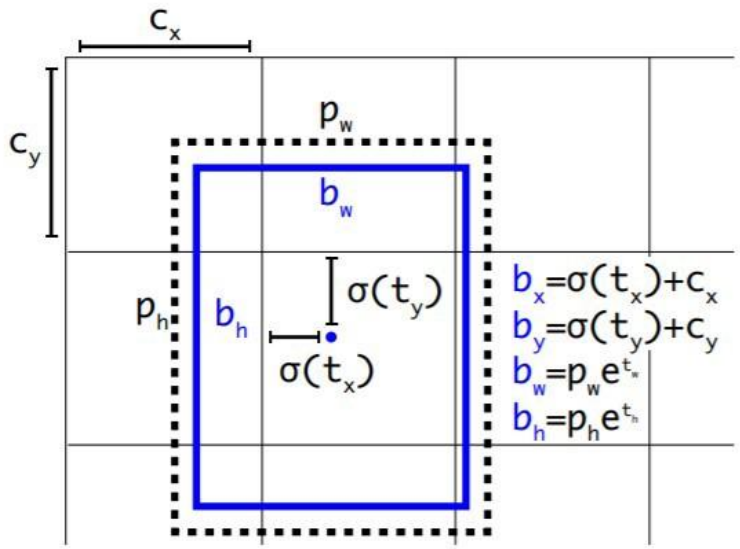

Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

Figura 4: Fórmulas para calcular las coordenadas de las bunding boxes a partir de los anchors y la salida de la red neuronal.

La salida de la red neuronal son tres grids, uno por escala, en el que cada celda del grid contiene tres vectores (uno por anchor) en el que cada vector tiene los datos que se observan en la Figura 5. Cada vector de cada anchor contiene las coordenadas de la bounding box en realción con un anchor que se usarán para calcular la bounding box real usando la fórmula de la FIgura 5. A parte también se proprciona un *object score* que indica la probabilidad de que esa *bounding box* contenga un objecto (a este score se le debe aplicar una *Sigmoide* para asegurar que su resultado esté entre 1 y 0). Finalmente el vector contiene una serie de elementos correspondientes con la probabilidad de que ese objeto peretenzca a esa clase. Estos últimos elementos puden ser tratados de dos formas, dependiendo de si una bounding box sólo puede ser una clase o pueden ser varias (entrenar usando las clases de ImageNet). En este caso se ha usado la aproximación de *Single Class Classification.*

Con todos estos datos se puede introducir la función de pérdida. En la Figura 5 se puede ver la forma de la función de pérdida orignal, pero en este caso se han cambiado las formas en que se calculan las pérdidas en cada apartado. En el caso de las coordenadas se ha usado *Mean Squared Error*, en el caso del score (tanto para objeto cómo para no objetos) se ha usado *Binary Cross Entropy* y en el caso de las clases *Cross Entropy*.

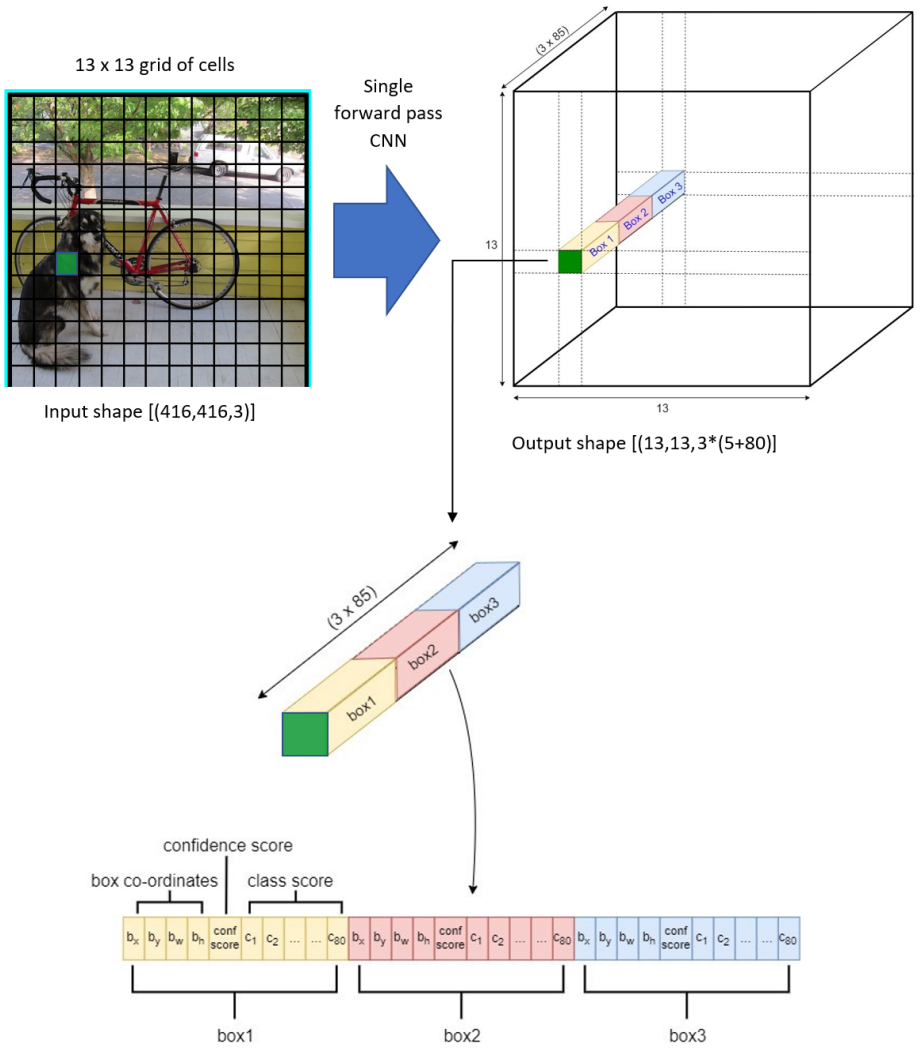

Figura 5: Salida de una celda de una escala de la red YoloV3

Regression loss

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Confidence loss

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

Classification loss

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Figura 6: Función de pérdida de Yolo

Entrenamiento

Para comprobar el estado del entrenamiento así como las predicciones de las imágenes se ha usado *Tensorboard*. Tanto la pérdida de validación cómo de entrenamiento se almacena en tensorboard. Todos estos datos estarán disponibles en la carpeta *fechaEjecución-logs* cuando se lanze un entrenamiento. Un ejemplo de predicción sin entrenamiento así cómo varias estadísticas de entrenamiento se pueden observar en la FIgura 7 y 8 respectivamente.

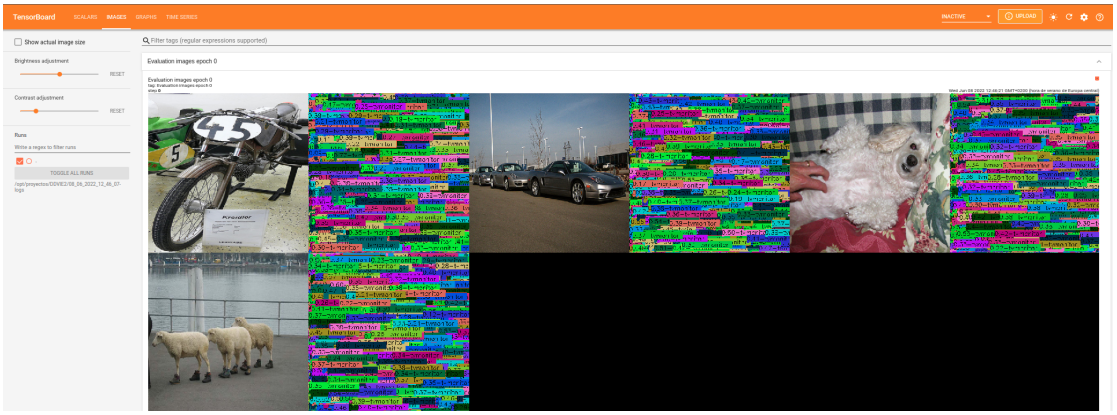

Figura 7: Predicciones sin entrenamiento en tensorflow

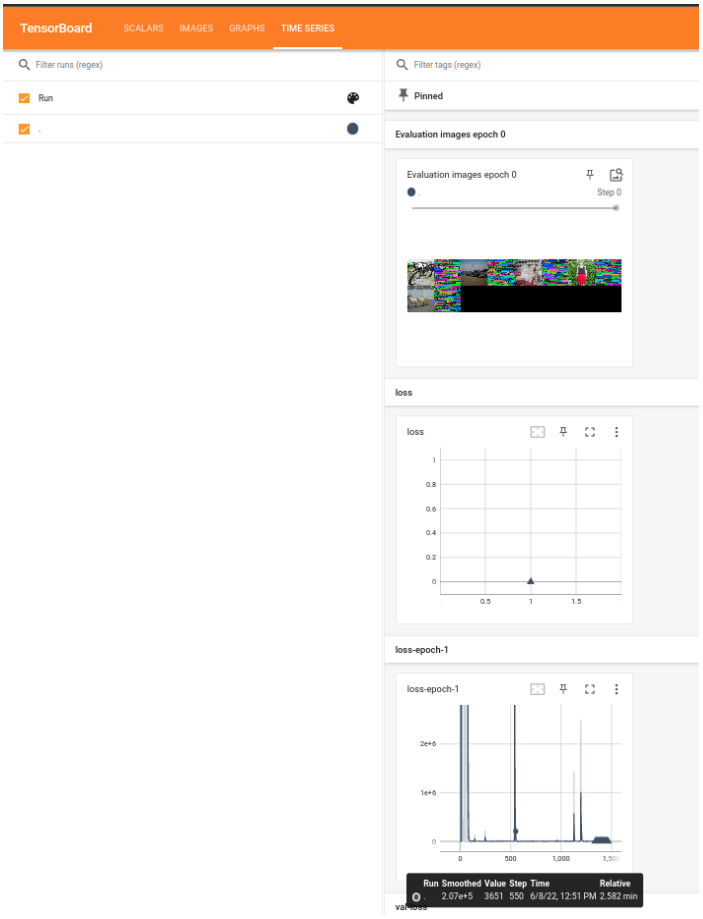

Figura 8: Estadísticas de tensorboard

La topología desarrollada mostrada en tensorboard se puede observar en la FIgura 9.

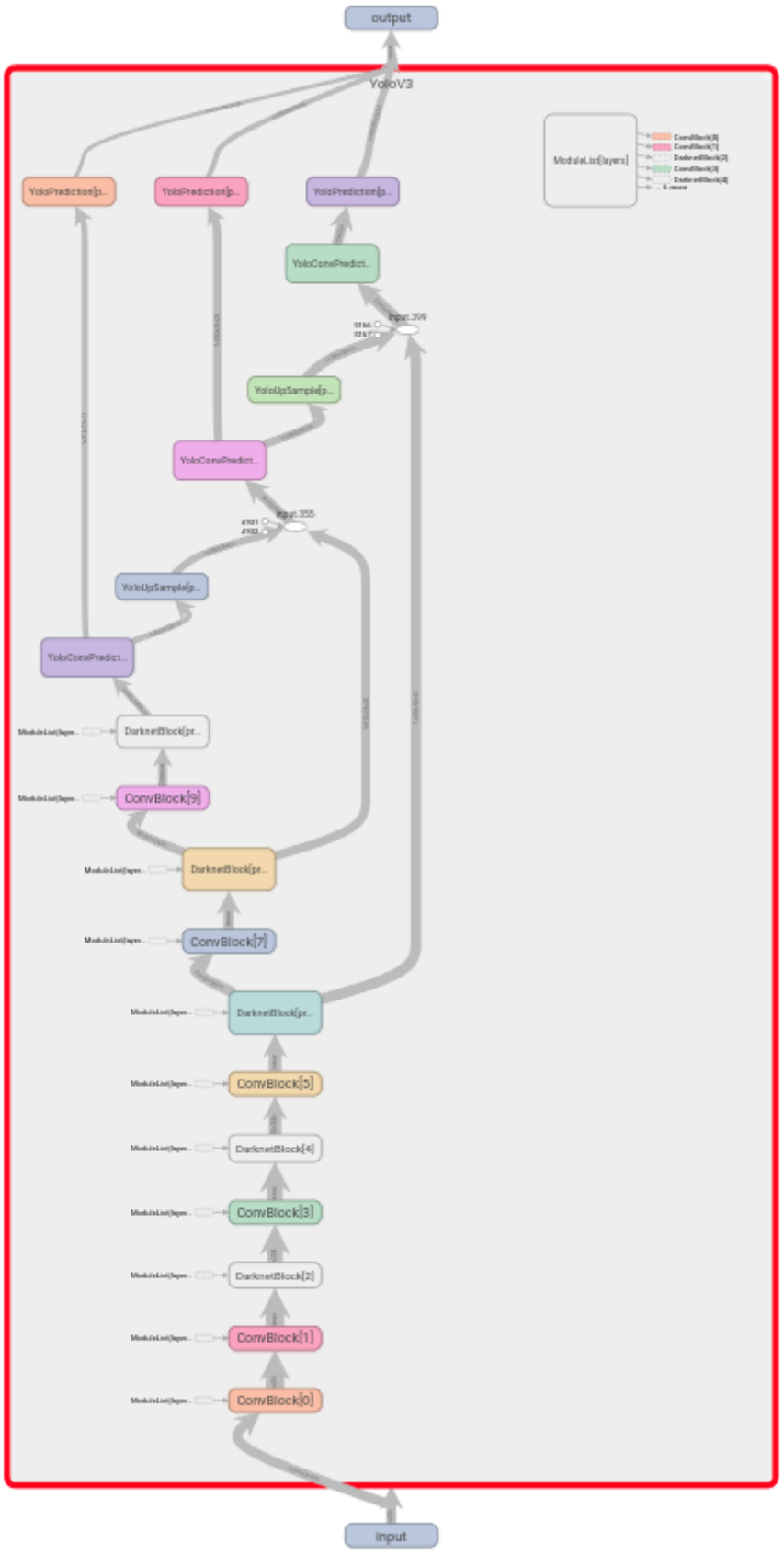

Figura 9: Topología YoloV3 desarrollada