Analysis: Focal Loss for Dense Object Detection (RetinaNet)

Aditya Kunapuli
Bourns College of Engineering
University of California, Riverside
Winston Chung Hall
446 N Campus Dr
Riverside, CA 92507
akuna002@ucr.edu

Abstract

This paper analyzes the novel implementation of a weighting factor to the loss function as outlined in the article "Focal Loss for Dense Object Detection"[1] by Lin et al. *The author's coin the term "focal loss" for this weighting factor and its purpose is to modulate the effects of class imbalances in context of single-stage detectors. The analysis portion concludes by demonstrating that the performance of focal loss exceeded not only the current leading single-stage detection algorithms, but also leading two-stage algorithms as well. The final section contains example images of the author's own implementation.*

1. Problem Statement

The primary motivation behind the introduction of focal loss is to improve performance of one-stage detectors, specifically in context of dealing with large class imbalances, a problem that two-stage detectors don't posses. The specifics of the two families of detection algorithm is explored in detail below.

1.1. Two Stage Detection

The problem of *object detection* fundamentally differs from *object classification* in that the purpose of the former is to detect the areas within an image that have a high probability of containing objects of interest.

The current state-of-the-art frameworks for object detection utilize a two stage process in which:

1. the first stage, utilizing a Region Proposal Network (RPN), generate sparse set of *region proposals* that cover Regions of Interest (ROI), and
2. the second stage, which utilizes a Convolutional Neural Network (CNN) to classify each proposal as either one of the foreground classes or as background.

The family of two-stage methodologies, such as R-CNN[2], Fast R-CNN[3], Faster R-CNN[4], Mask R-CNN[5] or some variants of these four archetypes, continue to demonstrate the highest average precision in COCO (AP COCO) evaluations[6].

Though the most immediately apparent side-effect of two-stage models and their associated AP, is that they are also computationally demanding. This can be attributed in large part to their requirement for very large input sizes, which in turns is a prerequisite for the ROI Pooling operations[7]. The end result is the first stage produces a large number of candidate boundary boxes for each image, that is then fed into a CNN for classification.

1.2. Single Stage Detection

In comparison, a new family of object detectors that prioritizes speed over accuracy, has emerged and quickly gained popularity. These models, called Single Shot Detectors (SSD[1]), combine the process of boundary box prediction and classification into a single stage.

Detector	Backbone	AP
YOLOv2	DarkNet-19	21.6
SSD513	ResNet-101-SSD	31.2
DSSD513	ResNet-101-DSSD	33.2

Table 1. Performance of various One-Stage Models. [8]

As described, these models prioritize speed over accuracy, as such with the resulting decrease in computation time and computational resources, comes reduced performance. Table 1 shows a performance comparison between the various Single Shot Detection models.[2].

[1] Not to be confused with the actual model itself, by the same name
[2] For a much more thorough survey in performance differences be-

Though nonetheless, there are many cases in which such a trade off is worth it, such as when used in consumer mobile devices.

A much more compelling problem arises in Single Stage Detectors–namely that of *class imbalance*. Specifically, the *intrinsic imbalance* that arises in naturally occurring frequencies of data[10]–*e.g.* medical diagnoses of cancer in a large population of healthy patients. In context of object detection models, Lin *et al.* summarized the problem as thus:

> These detectors evaluate $10^4 - 10^5$ candidate locations per image but only a few locations contain objects. [1]

The issue of class imbalances and their effects on back-propagation algorithms within shallow neural nets has been documented and studied since at least the early 1990s. Anand *et al.* showed that the effect of class imbalances led to the majority class dominating the net gradient[11]. Figure 1 demonstrates this–note that the error associated with the majority class quickly declines, whereas the error associated with the minority class essentially explodes until the model is *actually performing worse than simply flipping a coin*. Or to put it another way: as a result of severe class imbalances, the model shown in Figure 1 is performing so poorly in respect to the minority class, that actual prediction may be improved by reversing the assignment of probabilities (in a binary classification).

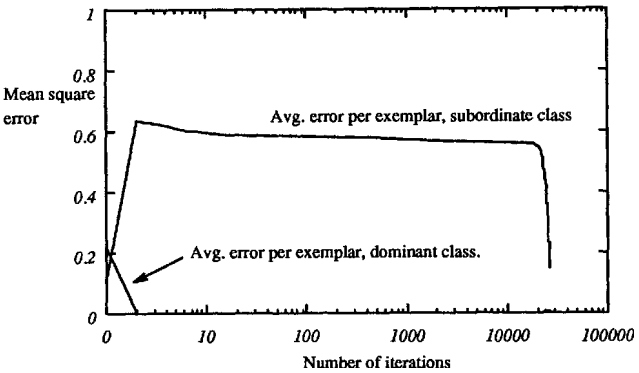

Figure 1. Effect of class imbalances. [11]

tween current architectures, see Ref [9]. Figure 10 at the end of this analysis shows one of the results of the survey.

2. Approach

2.1. Focal Loss

We begin by introducing the definition of cross entropy (CE) loss in the case of binary classification:

$$\mathrm{CE}(p,y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise} \end{cases} \tag{1}$$

Where $y \in \{\pm 1\}$ describes the ground truth class and $p \in [0,1]$ is the model's estimated probability for a given class with label $y = 1$. Rewriting the p equation such that

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases} \tag{2}$$

Allows us to rewrite CE such that:

$$\mathrm{CE}(p,y) = -\log(p_t) \tag{3}$$

To reiterate: the problem at hand is dealing with large classes imbalances in single step detectors. Such imbalance overwhelm the cross-entropy loss; easy to classify negatives (*i.e.* easy negatives) end up contributing the majority of the loss and dominate the gradient. In terms of medical diagnoses, this would be the equivalent of attempting to create a model for predicting cancer by studying a population of primarily cancer-free patients. In such scenarios, the easy negatives (i.e. cancer-free patients) would end up defining the model. Whereas, in reality we are truly interested in the population with cancer.

A method common in dealing with this class imbalance problem is the introduction of a weighting factor α to the cross-entropy loss function such that:

$$\alpha_t = \begin{cases} \alpha & \text{if } y = 1 \\ 1-\alpha & \text{otherwise} \end{cases} \tag{4}$$

Where $\alpha \in [0,1]$. This modification of CE is known as *balanced cross entropy*. The addition of this weighting factor helps to balance the respective importance of positive and negative training examples

Utilizing the aforementioned example of modeling cancer, the introduction of α would now allow us to increase contributions to the model from the minority class (*i.e.* patients with cancer), while simultaneously decreasing the role of the majority class.

And while this is a step in the right direction, α makes no additional distinction in the contributions from easy and hard examples.

To deal with this inability to distinguish easy and hard samples, the author's introduced a new factor in place of α called *focal loss*, which they defined as:

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t) \tag{5}$$

The coefficient $-(1-p_t)^{\gamma}$ is known as the *modulating factor*, where γ is a tunable *focusing hyperparameter*. Figure 2 shows how choice of γ can influence loss. Note that the case of $\gamma = 0$ (*i.e.* the top-most line in blue) corresponds to the normal cross entropy loss function.

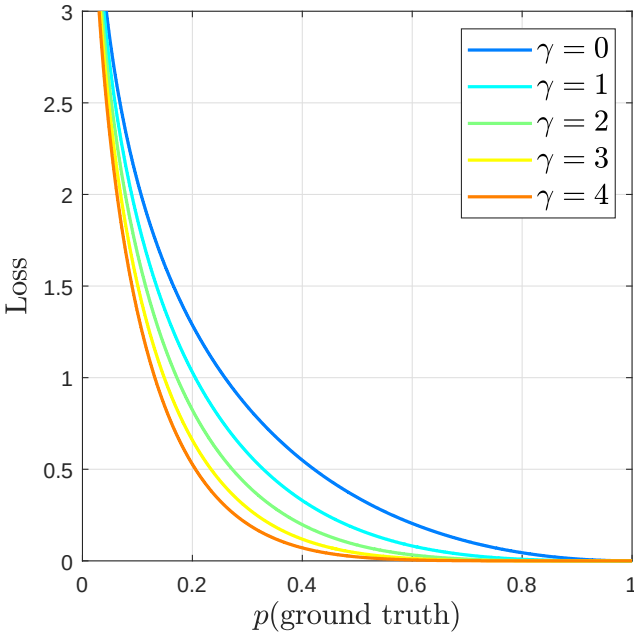

Figure 2. Loss for varying values of γ.

Revisiting equation (2) in conjunction with Figure 2 we can observe that the focal loss function has the following properties:

1. Misclassifications associated with a high probability assignment (*i.e.* if $y \neq 1$ and the associated p is large), result in the $(1-p_t) \to 1$ and hence the loss term remains unchanged.

2. Alternatively, in cases of correct classification with a high probability ($y = 1$ and $p \to 1$) or misclassifications with low probability both result in $p_t \to 1$ and hence $(1-p_t) \to 0$ and the loss contribution for that example is reduced.

3. The focusing parameter γ smoothly adjusts the rate at which easy example's loss contribution is reduced.

A summary of the various effects to the loss function is provided in Table 2.

	High p	**Low** p
Correct	Reduce Loss	Loss Unchanged
Incorrect	Loss Unchanged	Reduce Loss

Table 2. Changes to loss

3. Experimental Setup

The author's implemented the focal loss function into a Feature Pyramid Network (FPN) backbone, which itself rests upon the ResNet architecture[12]. As the backbone itself is, in the author's own words. an *"off the shelf convolutional network"*, we're not going to cover any additional detail to that end. The model was trained with Stochastic Gradient Descent (SGD) against the COCO `trainval135k dataset`, utilizing 80k images for training and a random sample of 35k images for testing pulled from the validation set.

4. Analysis

Figure 3 shows the results of varying γ on cumulative loss for positive and negative samples, once the model had converged. The effects of changing γ for the positive samples was minor, though for the negative samples, greater values of γ had the effect of focusing the accumulated loss on the the hard examples–*i.e.* severely reducing the contribution of easy negatives. These finding match the predictions made in Table 2.

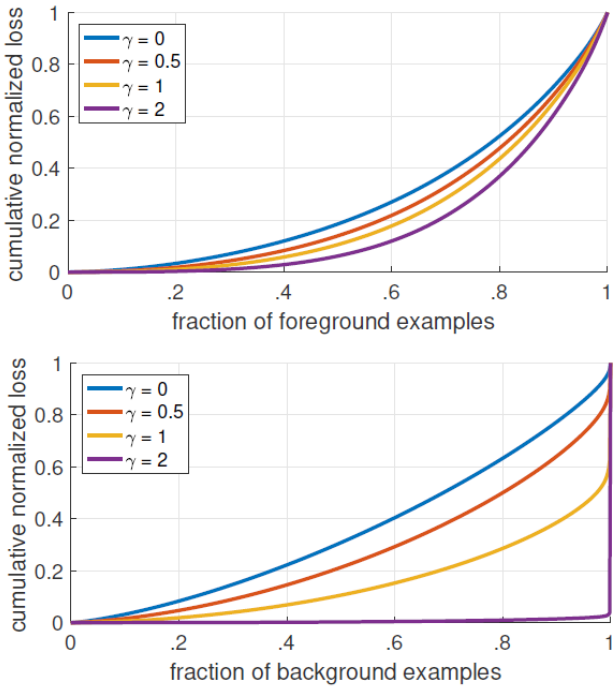

Figure 3. Accuracy vs time of various architectures. [9]

The author's additionally note that the precise form function used for the modulating factor is not critical. They demonstrate this by testing focal loss with the following modulating factor:

$$p_t^2 = \sigma(\gamma x_t + \beta) \tag{6}$$

$$FL^* = -\frac{1}{\gamma}\log\left(p_t^*\right) \tag{7}$$

The differences between FL and FL^* are trivially minor as shown in Table 4:

Loss	γ	β	AP
CE	–	–	31.1
FL	2.0	–	34.0
FL^*	2.0	1.0	33.8
FL^*	4.0	0.0	33.9

Table 3. Results of FL and alternative, FL^*. [8]

5. Results

In terms of model performance, RetinaNet managed to outperform not only other Single-Shot Detectors, but leading (at the time) Two-Stage Detectors, seen here in Table 4. Note that since the introduction of this model in late

Detector	Backbone	AP
Two Stage Detectors		
Faster R-CNN+++	ResNet-101-C4	34.9
Faster R-CNN-FPN	ResNet-101-FPN	36.2
Faster R-CNN (G-RMI)	Inception-ResNet-v2	34.7
One Stage Detectors		
YOLOv2	DarkNet-19	21.6
SSD513	ResNet-101-SSD	31.2
DSSD513	ResNet-101-DSSD	33.2
RetinaNet	**ResNet-101-FPN**	**39.1**
RetinaNet	**ResNeXt-101-FPN**	**40.8**

Table 4. Performance of Two-Stage and One-Stage Models. [8]

2017, various others have surpassed it (*e.g.* MegData[6]) as the high score in COCO leaderboards continues to trend up. though RetinaNet and its variant continue to enjoy widespread success as evidenced by the nearly 1,000 citations to it[1].

Figure 4. Number of citations to the original paper. [9]

6. Example Images

As I wasn't able to implement a complete training cycle from scratch, I chose instead to utilize Keras/Tensorflow's provided examples [13] to produce the following images. The images (from my own collection) demonstrate the models ability to identify and classify objects.

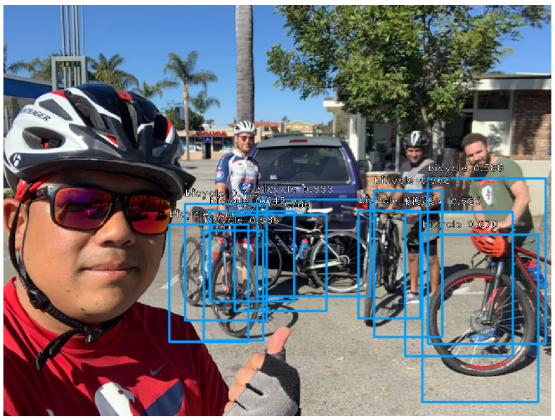

Figure 5. Bounding boxes for bicycles.

Figure 6. Bounding boxes for a car.

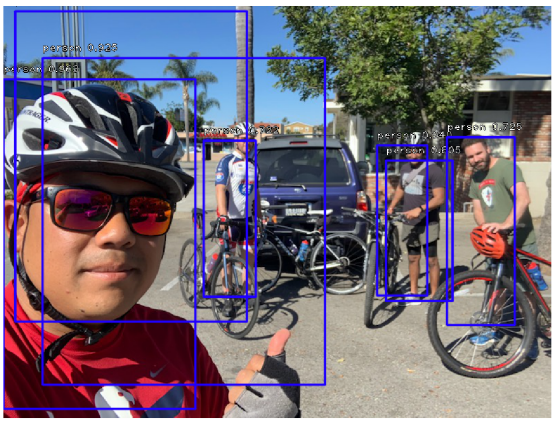

Figure 7. Bounding boxes for people.

The model utilized pre-trained weights and a backbone

of ResNet50 (as seen in Table 4, a comparatively more advanced backbone model would performed even better). The code used in the generation of these images is included in the file with this analysis.

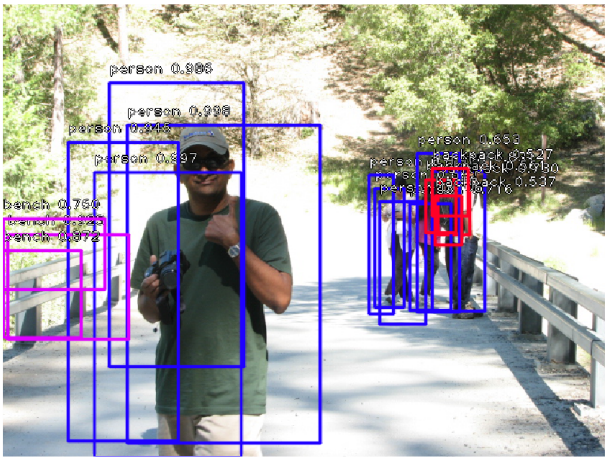

Figure 8. Bounding boxes around people, benches and backpacks.

References

[1] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[2] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[3] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[4] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[5] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[6] "Common objects in context," 2019. [Online]. Available: http://cocodataset.org/#detection-leaderboard

[7] B. Cheng, Y. Wei, H. Shi, R. S. Feris, J. Xiong, and T. S. Huang, "Revisiting RCNN: on awakening the classification power of faster RCNN," *CoRR*, vol. abs/1803.06799, 2018.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[9] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[10] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," Mar 2019. [Online]. Available: https://link.springer.com/article/10.1186/s40537-019-0192-5

[11] R. Anand, K. G. Mehrotra, C. K. Mohan, and S. Ranka, "An improved algorithm for neural network classification of imbalanced training sets," *IEEE transactions on neural networks*, vol. 4 6, pp. 962–9, 1993.

[12] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[13] Fizyr, "fizyr/keras-retinanet." [Online]. Available: https://github.com/fizyr/keras-retinanet/blob/master/examples/ResNet50RetinaNet.ipynb

7. Additional Figures

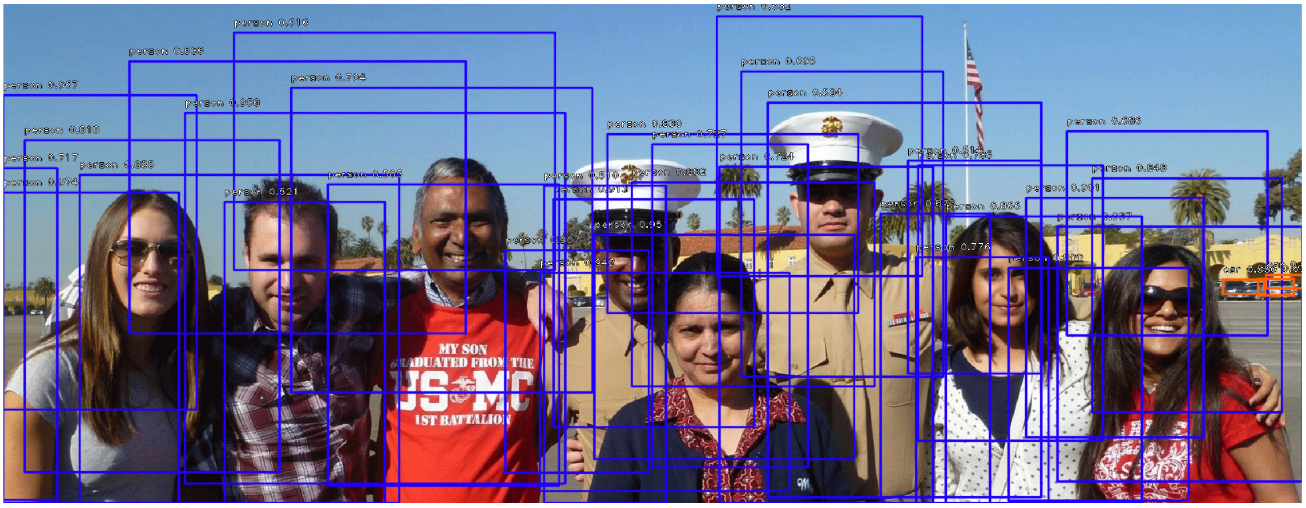

Figure 9. Bounding boxes around people.

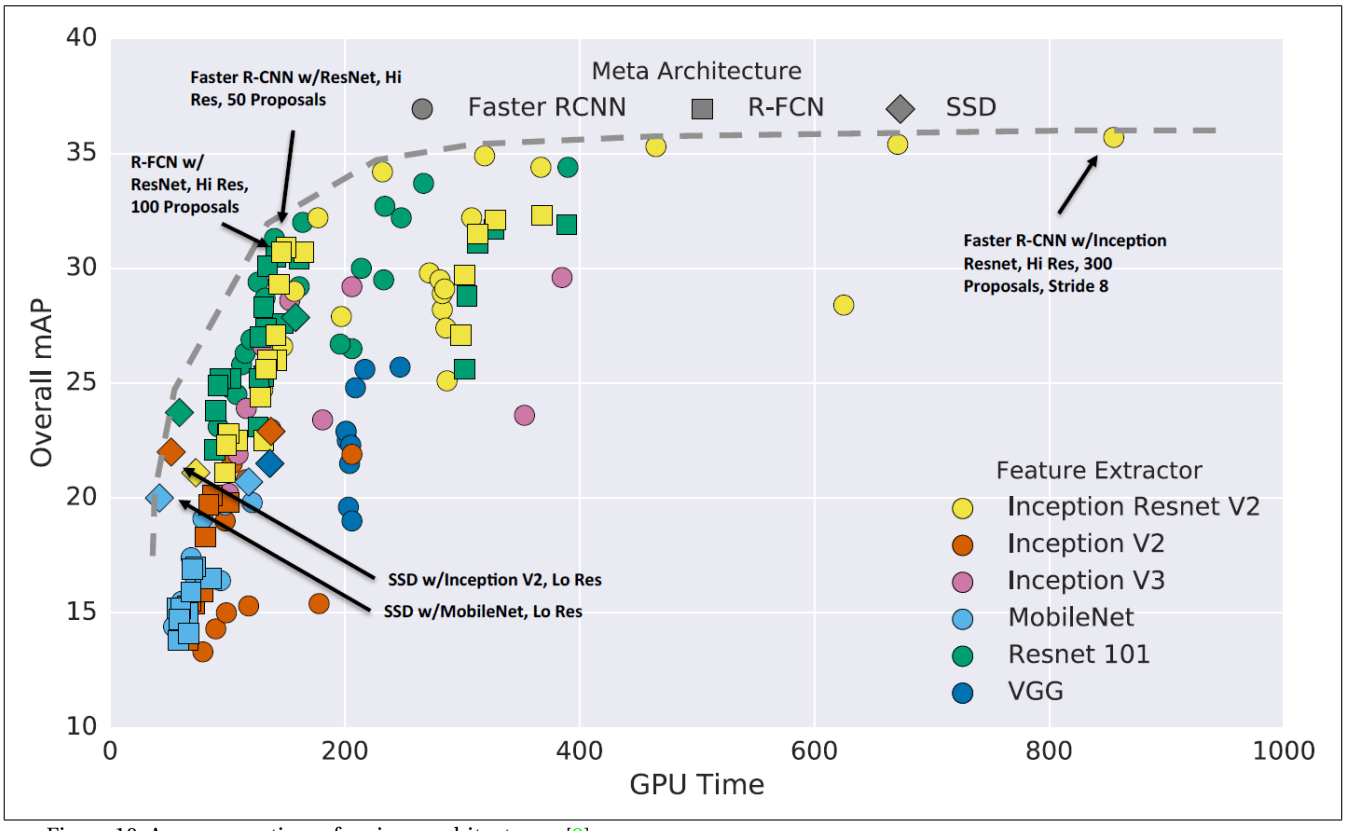

Figure 10. Accuracy vs time of various architectures. [9]