Variational Feature Pyramid Networks

Panagiotis Dimitrakopoulos [1] **Giorgos Sfikas** [1 2 3] **Christophoros Nikou** [1]

Abstract

Recent architectures for object detection adopt a Feature Pyramid Network as a backbone for deep feature extraction. Many works focus on the design of pyramid networks which produce richer feature representations. In this work, we opt to learn a dataset-specific architecture for feature pyramid networks. With the proposed method, the network fuses features at multiple scales, it is efficient in terms of parameters and operations, and yields better results across a variety of tasks and datasets. Starting by a complex network, we adopt Variational Inference to prune redundant connections. Our model, integrated with standard detectors, outperforms the state-of-the-art feature fusion networks.

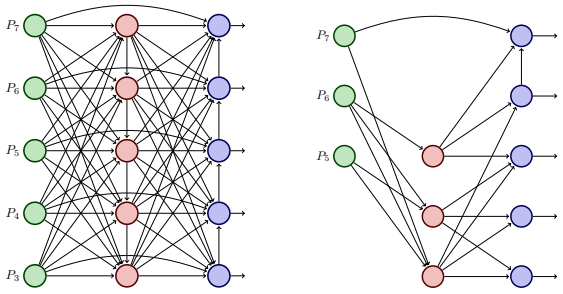

Figure 1. An illustration of pruning under the proposed method. The image on the left depicts the initial model before training which is highly complex, including multiple-level feature fusion and is "fully" connected. On the right, the same network is shown after 10 epochs of training, where redundant connections and building blocks have been pruned, leading to an efficient fusion network.

1. Introduction

Object detection and instance segmentation are two of the most fundamental problems in the computer vision field. These problems are quite difficult to solve and provide multiple challenges as multiple objects have to be detected or segmented at multiple scales, locations and under different conditions. The majority of those problems in practice are today approached via the use of deep learning architectures. In the recent years multiple deep architectures have been proposed, leading to widely known models in computer vision (Ren et al., 2016; He et al., 2017; Redmon et al., 2016; Carion et al., 2020). Neural network-based detectors are typically built and designed upon deep robust feature extraction (sub-)networks referred to as backbones. The task of these components is to transform the input image to a deep embedded representation, subsequently fed to the detector head in order to have it produce the required predictions.

These networks heavily rely on good representations, as the input feature must be descriptive enough to capture the different relations and scales among the image objects. A common practice is to use pretrained deep convolutional backbones such as ResNet (He et al., 2016) or Inception (Szegedy et al., 2015) to first extract rich features at different scales from the input image. Subsequently, their output is processed using a pyramidal-shaped multi-scale fusion network in order to create richer and more descriptive features. Since the introduction of Feature Pyramid Networks (FPNs) (Lin et al., 2017a), which standardize the above procedure, works have been proposed to design more flexible, sophisticated and also efficient fusion networks.

In the current work, we are using Variational Inference (VI) in order to build more sophisticated feature fusion networks that can be used for detection tasks. The model we propose is applicable in a generic deep-learning based detection context. We combine multiple parts of state-of-the-art fusion networks to create an "initial" complex network, that is subsequently pruned to its more efficient counterpart. In order to obtain the pruned network, we opt to learn to highlight the network components that are more suitable for the specific task and dataset that it is trained on. Numerical experiments show that the models produced using the proposed method, combined with various object detectors, produce state-of-the-art results on a variety of detection tasks.

[1]Department of Computer Science and Engineering, University of Ioannina, Ioannina, Greece [2]University of West Attica, Athens, Greece [3]National Center for Scientific Research "Demokritos", Athens, Greece. Correspondence to: Panagiotis Dimitrakopoulos <p.dimitrakopoulos@uoi.gr>, Giorgos Sfikas <sfikas@cse.uoi.gr>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

We begin in Section 2 with a discussion of previous work with respect to two key aspects related to our contribution: FPNs and variational methods for pruning deep neural networks. In Section 3, we formally define the proposed variational FPN and its components, and analyze how parameters can be interpreted as stochastic random variables. We show an efficient way to apply inference on those parameters, and how certain parts of the network can be pruned with an imposed sparse prior. We evaluate the proposed approach using numerical experiments in Section 4. The paper is concluded with a discussion on our contribution and future work in Section 5.

2. Related Work

2.1. Feature Pyramid Networks

Feature Pyramid Networks (Lin et al., 2017a) were proposed as an architectural solution to providing a multi-scale feature representation of the input. A "pyramidal"-structured hierarchy of feature maps is to be produced by a convolutional pipeline and combined to high-level semantic outputs. A bottom-up and a top-down pathway are the basic structural sub-elements of the feature pyramid. The bottom-up pathway computes a feature hierarchy, where each level corresponds to a different resolution scale. On the top-down pathway, starting from the coarsest resolution towards the finest one, feature maps are progressively upsampled (e.g. by a constant factor of 2), and combined with corresponding bottom-up maps. Over each pair of top-down and bottom-up corresponding blocks, the top-down map is semantically high-level, and the bottom-up map is semantically low-level.

There are a lot of recent works that propose sophisticated modules to extract more representative features for object detection tasks. Telling examples include (Wang et al., 2020), where the Pconv module is introduced to simultaneously extract features at different scales; attention-based modules (Kong et al., 2018; Dai et al., 2021); or even methods that discard the whole FPN structure (Chen et al., 2021) and methods that incorporate the popular Transformer architecture (Wang et al., 2021). PANet (Liu et al., 2018) adds an extra bottom-up pathway on top of the original FPN architecture. The M2Det object detector (Zhao et al., 2019b) extends the idea and builds stronger feature pyramid representations by employing multiple U-shape modules after backbone pipeline.

Another approach, more related to our method, is NAS-FPN (Ghiasi et al., 2019). This approach uses a Neural Architecture Search (NAS) algorithm to find an optimal structure instead of manually designing architectures for pyramidal representations. This model requires a significant computational load for training, and the output network is irregular and difficult to interpret or modify. In (Tan et al., 2020), the proposed BiFPN model uses less building blocks as the authors drop blocks with only one input feature map. Furthermore, they add an extra edge linking the original input to an output node if they are at the same level, in order to fuse more features while avoiding too much extra cost. Another novelty of BiFPN is weighted fusion, with which they introduce a set of learnable parameters associated with each input feature map on every block. In this manner, the network is allowed to learn the importance of each separate feature map.

2.2. Variational Pruning Methods

The proposed method can be viewed as a specific case, suitable for deep object detectors, of pruning neural network parameters. The notion of pruning parameters of deep neural networks comes to relax the implementation difficulties on resource-constrained platforms. A successful pruning method must be able to compress the model and improve efficiency with a minimal loss in terms of accuracy. To this end, probabilistic approaches using Variational Inference have already been deployed. (Kingma et al., 2015) treat weights of neural networks as random variables and leverage Variational Inference to efficiently estimate the parameters, in a model that is shown to elegantly generalize Gaussian dropout (Srivastava et al., 2014). (Molchanov et al., 2017) revised the previous work and proposed a scheme to estimate the dropout rate, proving that the resulting method leads to sparse solutions. Further works, imposing sparse priors on the weights (Louizos et al., 2017), proposed the use of hierarchical priors on hidden units; on a different note, neurons can be pruned altogether, including all their incoming and outgoing weights. This avoids more complicated and inefficient encoding schemes. Unfortunately, these methods cannot be generalized to complex convolutional layers found in modern deep learning models due to the complexity and interdependence of operations. To tackle this problem, more general probabilistic methods of network pruning have been proposed. In (Zhao et al., 2019a) for example, the batch normalization layer is reformulated, where the normalized features are multiplied channel-wise with a sparse prior-based stochastic parameter that effectively prunes redundant channels.

3. Proposed Model

We now formally introduce our multi-scale feature fusion network that combines aspects proposed in recent works such as (Tan et al., 2020), which we will use as a network baseline. Subsequently, instead of using deterministic fusion weights, we show how to estimate the distribution of fusion weights cast as random variables via the use of Variational Inference. We analytically describe how the Stochastic Gradient Variational Bayes (SGVB) estimator (Kingma &

Welling, 2014) can be used to apply inference on the fusion weights. Finally, we introduce sparse prior distributions like Automatic Relevance Determination (ARD) to effectively prune network connections, thus obtaining a model with the optimal lower complexity.

3.1. Multi-Scale Feature Fusion Network

The architecture of our network is initialized as a 'fully connected' version of the PaNET (Liu et al., 2018) (with bottom-up and top-down pathways) using skip connections. On each building block, the input features are combined via the use of fast normalized fusion, an efficient approximation of softmax proposed in (Tan et al., 2020). An illustration of this initial architecture is depicted in Figure 1 (left).

The output of a building block F_{out} in our network is formally defined as follows. Let $F_{level}^{layer} = \{F_1, \ldots, F_N\}$ be the set of all N features that are input to the given block and $W_{level}^{layer} = \{w_1, \ldots, w_N\}$ a set of weights $w_i \geq 0$ each of which is associated with one input feature from F_{level}^{layer}. Then for F_{out} we write:

$$F_{out} = \text{Conv}(\frac{\sum_{i=1}^{N} w_i F_i}{\sum_{i=1}^{N} w_i + \epsilon}), \tag{1}$$

where ϵ is a small constant added for numerical stability and Conv stands for a 2D convolutional operation. Note that all features in F_{level}^{layer} are resized using bilinear interpolation when they correspond to cross-scale connections. All building blocks are considered to be identical.

3.2. Variational Inference

In our method, we treat each weight $\mathbf{w}$ associated with each connection on the network as a stochastic variable coming from a parametric distribution $p(\mathbf{W})$. We consider a detection or segmentation dataset $D = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{J}$ as a set of random variables, where $\mathbf{x}$ is input data and $\mathbf{y}$ is the corresponding ground truth, in a dataset comprised of J images. The joint distribution which combines and depicts the relations of model variables is defined in the following way:

$$p(\mathbf{X}, \mathbf{Y}, \mathbf{W}) = p(\mathbf{Y}|\mathbf{X}, \mathbf{W})p(\mathbf{W}). \tag{2}$$

Our goal is to estimate the posterior distribution of latent variables $\mathbf{W}$ i.e. $p(\mathbf{W}|\mathbf{Y}, \mathbf{X})$. Since the posterior distribution cannot be obtained in closed form, we cannot apply exact inference methods, thus we resort to approximate inference and specifically to the variational Bayesian methodology (Bishop, 2006). We assume a family of approximate posterior distributions $q_\phi(\mathbf{W})$ parameterized by ϕ, and then seek values for the parameters ϕ that best approximate the true posterior. Model evidence $p(D) = \int p(D, W)dW$ can be decomposed into:

$$\log p(D) = \mathcal{L}(\mathbf{W}) + KL(q_\phi(\mathbf{W})||p(\mathbf{W}|D)), \tag{3}$$

where $\mathcal{L}(\mathbf{W})$ is the Variational Lower Bound (VLB) and $KL(q_\phi(\mathbf{W})||p(\mathbf{W}|D))$ is the Kullback-Leibler (KL) divergence between the distribution $q_\phi(\mathbf{W})$ and true posterior distribution $p(\mathbf{W}|D)$. The best approximation of the true posterior distribution comes via maximizing the lower bound, a process which is equivalent to minimizing the KL divergence. Unfortunately, the required integrals for applying a mean-field VB algorithm are also intractable. These intractabilities appear since the detection networks are extremely complicated likelihood functions $p(\mathbf{Y}|\mathbf{X}, \mathbf{W})$. Thus, in order to find the posterior distribution w.r.t. hidden variables we directly optimize the VLB, which we write as:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}_{q_\phi(\mathbf{W})}[\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W})] - KL(q_\phi(\mathbf{W})||p(\mathbf{W})). \tag{4}$$

We want to differentiate and optimize the VLB $\mathcal{L}(W)$ w.r.t. both the variational parameters ϕ and generative parameters θ. However, the gradient of the VLB w.r.t. ϕ is not trivial to compute. We proceed by using SGVB (Kingma & Welling, 2014) and use an estimate $\tilde{\mathcal{L}}(\mathbf{W}) \simeq \mathcal{L}(\mathbf{W})$ of the lower bound and its derivatives w.r.t. the parameters. According to SGVB, the approximate posterior $q_\phi(\mathbf{W})$ can be reparameterized via a differentiable transformation $f(\phi, \epsilon)$ of an (auxiliary) noise variable ϵ:

$$\mathbf{w} = f(\phi, \epsilon), \quad \text{where} \quad \epsilon \sim p(\epsilon) \tag{5}$$

We can now form a low-variance Monte Carlo estimate on the expectation appearing in (4). Under the change-of-variables rule for integrals, the expected log-likelihood is the same as the expectation w.r.t. the auxiliary distribution

$$\begin{aligned} \tilde{\mathcal{L}}(\mathbf{W}) = \frac{1}{L} \sum_{l=1}^{L} \sum_{j=1}^{J} \log p(y_j|x_j, \mathbf{W} = f(\mathbf{w}, \epsilon_{l,j})) \\ - KL(q_\phi(\mathbf{W})||p(\mathbf{W})), \end{aligned} \tag{6}$$

where $\epsilon_{l,j}$ is the l^{th} sample of $p(\epsilon)$ for the j^{th} input datum.

3.3. Choice of Prior Distribution

Here we explicitly define the prior distribution for the connection weights, the choice of the approximate variational posterior distribution and the VLB. Regarding the type of prior distribution we have to account for several things. First, since we want the fusion architecture to be as far from complex as possible we need our model to discard redundant connections –and if possible entire building blocks– that do not contribute to the model accuracy. Thus, we have to choose a sparse prior that will gear a significant part of connections towards zero. Also, for the SGVB to be applicable, the distribution must be reparameterized w.r.t. an auxiliary variable, and finally the KL term must be easy to compute while also being numerically stable in order to facilitate efficient training.

3.3.1. Automatic Relevance Determination

The mechanism of Automatic Relevance Determination (ARD) is a well studied subject, first introduced in the context of sparse linear regression using relevance vector machines (Tipping, 1999; Titsias & Lázaro-Gredilla, 2014). This setting causes a subset of parameters to be driven to zero. Assuming that the weights are independent and identically distributed (i.i.d.) we set the prior distributions to zero-mean Gaussian [1]:

$$p(\mathbf{W}) = \prod_i p(\mathbf{w}_i) \text{ where } \mathbf{w}_i \sim \mathcal{N}(0, \hat{\sigma}_i^2) \tag{7}$$

The straightforward choice for the approximate variational distribution that satisfies the conditions for applying SGVB is the factorized Gaussian:

$$q(\mathbf{W}) = \prod_i q(\mathbf{w}_i) \text{ where } \mathbf{w}_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{8}$$

The set of variational parameters to be optimized is $\phi = \{\mu, \sigma\}$. After reparameterization, we have:

$$\mathbf{w} = \mu + \sigma\epsilon \text{ where } \epsilon \sim \mathcal{N}(0, 1). \tag{9}$$

The optimal hyperparameter $\hat{\sigma}$ of the prior distribution can be calculated by optimizing the VLB:

$$\frac{\partial \tilde{\mathcal{L}}(\mathbf{W})}{\partial \hat{\sigma}_i^2} = 0 \Rightarrow -\frac{\partial}{\partial \hat{\sigma}_i^2} KL(q_\phi(\mathbf{W})||p(\mathbf{W})) = 0, \tag{10}$$

which yields the optimal parameters $\hat{\sigma}_i^2 = \mu_i^2 + \sigma_i^2$. By substituting those parameters the KL term of the VLB can be computed analytically, as it is defined over Gaussian terms:

$$KL(q_\phi(\mathbf{W})||p(\mathbf{W})) = \frac{1}{2}\sum_i \log(\frac{\mu_i^2}{\sigma_i^2} + 1) \tag{11}$$

3.3.2. ARD with correlated weights

We extended the mechanism of Automatic Relevance Determination (ARD) in order to study the correlation between the connection weights and how it affects the pruning parameters of our method. We now set the prior distributions to zero-mean multivariate Gaussian:

$$p(\mathbf{W}) = \mathcal{N}(\mathbf{w}|0, \hat{\Sigma}), \tag{12}$$

where $\mathbf{w}$ is now a vector containing all the connection weights of our model. The straightforward choice for the approximate variational distribution that satisfies the conditions for applying SGVB is Gaussian:

$$q(\mathbf{W}) = \mathcal{N}(\mathbf{w}|\mu, \Sigma), \tag{13}$$

and the set of variational parameters that we wish to optimize is now $\phi = \{\mu, \Sigma\}$. Reparametrizing, we have:

$$\mathbf{w} = \mu + L\epsilon \text{ where } \epsilon \sim \mathcal{N}(0, I) \tag{14}$$

where $\Sigma = LL^T$ is the Cholesky decomposition of the covariance matrix Σ. The optimal hyperparameter $\hat{\Sigma}$ can be calculated directly by maximizing the VLB which yields parameters $\hat{\Sigma} = \mu\mu^T + \Sigma$. By substituting these parameters, the KL term of the VLB can be computed analytically. For numerical stability, the variational parameters that we estimate are the mean of the variational distribution μ and the lower triangular matrix with positive diagonal elements L^{-1}. Using the estimate matrix L^{-1}, we can sample from the variational distribution with eq. 14. Instead of inverting L^{-1}, we directly sample the weights by solving the triangular linear system $L^{-1}(\mathbf{w} - \mu) = \epsilon$. Solving the linear system is much more numerically stable and can be executed fast via hardware acceleration.

In order to avoid the inversion of the covariance matrices and the calculation of the determinants during the optimization procedure we decompose the KL term as:

$$\begin{aligned} &KL(q(\mathbf{W})||p(\mathbf{W})) \\ &= -\int q(\mathbf{w}) \log p(\mathbf{w}) d\mathbf{w} + \int q(\mathbf{w}) \log q(\mathbf{w}) d\mathbf{w} \\ &= -\mathbb{E}_{q(\mathbf{w})}(\log p(\mathbf{w})) - \mathcal{H}(q(\mathbf{w})), \\ &= \frac{1}{2}\log(|\hat{\Sigma}|) + \frac{1}{2}\mathbb{E}_{q(\mathbf{W})}(\mathbf{w}^T\hat{\Sigma}^{-1}\mathbf{w}) - \frac{1}{2}\log(|\Sigma|) + C, \end{aligned} \tag{15}$$

where $\mathcal{H}(q(\mathbf{W}))$ is the entropy of the variational distribution. By substituting the optimal hyperparameters found as $\hat{\Sigma} = \mu\mu^T + \Sigma$, the term $\log(|\hat{\Sigma}|)$ can be decomposed further by leveraging the matrix determinant lemma (Petersen et al., 2008):

$$\log(|\hat{\Sigma}|) = \frac{1}{2}\log(1 + \mu^T\Sigma^{-1}\mu) + \frac{1}{2}\log(|\Sigma|). \tag{16}$$

The second term in equation (15) can be computed using the reparameterization trick (14) which leads to the final expression for the KL term:

$$\begin{aligned} KL(q(\mathbf{W})||p(\mathbf{W})) = &\frac{1}{2}\log(1 + \mu^T\Sigma^{-1}\mu) \\ &+ \frac{1}{2}\mathbb{E}_{q(\epsilon)}(\tilde{\mathbf{w}}^T\hat{\Sigma}^{-1}\tilde{\mathbf{w}}), \end{aligned} \tag{17}$$

where $\tilde{\mathbf{w}}$ are the reparameterized sampled values for the weights using ϵ and the matrix $\hat{\Sigma}^{-1}$ can be easily computed via the Sherman-Morrison identity. This decomposition of the KL term allows us to avoid the inversion and log determinant operation in the VLB computations leading to very stable training of the network.

[1] In the remainder of the text, we omit the upper limit on i-indexed sums and products for brevity. This will be implied equal to the total number of network weights-connections, unless stated otherwise.

4. Experimental Results

In this Section, we provide numerical results for the proposed method, in comparison to recent existing feature fusion networks. For our numerical analysis, we perform three different experiments. First, we evaluate our methods as a backbone network for detection, using (Ren et al., 2016) and instance segmentation, using (He et al., 2017) versus state-of-the-art backbone combinations. We carried out experiments to evaluate how the learned architecture of our network can adapt to different types of datasets, containing objects at various scales and sizes. Furthermore, we tested the proposed probabilistic pruning methods versus different deterministic ones and we highlight the benefits of pruning components probabilistically. Finally, we introduce a way of acquiring uncertainty estimates of the model predictions and we evaluate the quality of those estimates experimentally.

4.1. Implementation Details

4.1.1. Model Details

Following (Lin et al., 2017b; Tian et al., 2019) we use feature pyramid levels P_3 to P_7, where P_3 to P_5 are computed from the output of the corresponding ResNet-50 residual stage (C_3 through C_5) using top-down and lateral connections, P_6 is obtained via a 3×3 stride-2 convolution on C_5, and P_7 is computed by applying ReLU followed by a 3×3 stride-2 convolution on P_6. These minor modifications have a positive impact on training and inference speed while maintaining accuracy. We used a 3×3 depth-wise separable convolution (Chollet, 2017) for feature fusion, as it reduces significantly the number of trainable parameters, and we added batch normalization and ReLU activation after each convolution. Each connection weight was restricted to positive values via the use of ReLU activation. Also, in order to avoid numerical instabilities our network optimizes the logarithmic variance $log(\sigma_i^2)$ of the variational distribution, instead of the equivalent optimization over σ_i^2. All the means were initialized with a value of 1 and the logarithmic variances were set to 0 corresponding to variance of 1. In order to improve training stability and also force our model to take advantage of all 5 levels, we added residual connections from P_3, P_4, P_5, P_6, P_7 to their respected outputs. This significantly helped the optimization procedure and slightly boosted model accuracy.

4.1.2. Training Details

Our main experiments are conducted on the large-scale detection benchmark COCO (Lin et al., 2014). Following common practices (Lin et al., 2017b; Tian et al., 2019), we use the COCO trainval35k split (115K images) for training and the minival split (5K images) as validation. We report our main results on the test dev split (20K images) by uploading our detection results to the evaluation server. Our model implementations were based on the MMDetection open source project (Chen et al., 2019). At each trial, we have trained the network for 15 epochs using Stochastic Gradient Descent (SGD) with momentum set to 0.9 and weight decay parameter set to 0.0001. The learning rate was set according to the linear scaling rule (Goyal et al., 2017); this rule states that the learning rate has to be proportional to the batch size, where each batch was set to include 2 input images, each at resolution of 1333×800 pixels. The training and test parameters for the employed detectors were set according to the values prescribed in the respective papers.

4.1.3. Variational Details

We prune redundant connections based on the distribution of weights w. When the means of the variational distribution are less than a threshold value, those connections are dropped; additionally, when a building block is left with zero input connections, the whole block is dropped, further reducing model complexity. Through our experiments, the KL term in the loss function was annealed by a small factor to avoid over-regularization. At test time, we followed the weight scaling rule (Srivastava et al., 2014) by replacing the weights with their expected values, i.e. formally:

$$\mathbb{E}_{q_\phi(\mathbf{w})} p(\mathbf{y}|\mathbf{x}, \mathbf{w}) \approx p(\mathbf{y}|\mathbf{x}, \mathbb{E}_{q_\phi(\mathbf{w})}[\mathbf{w}]). \tag{18}$$

4.2. Results

For fair comparison we followed the same training scheme for all the networks. As recent works have shown, repeating the same feature fusion network multiple times enables higher-level feature fusion and provides better accuracy. However, in our experiments we choose not to repeat the networks, as we believe that stacking layers makes the results of the experiments more difficult to interpret.

Table 1 shows the accuracy and model complexity for our proposed network and other state-of-the-art feature fusion networks, (NAS-FPN) (Ghiasi et al., 2019),(PANet) (Liu et al., 2018), (BiFPN) (Tan et al., 2020), (PConv) (Wang et al., 2020), (HRNet) (Sun et al., 2019). The variational-based networks were compared according to the choice of prior distribution. As we can see, in all cases accurate detection is quite a challenging problem. It is clear that supervised segmentation techniques like deep convolutional neural networks can benefit from the presence of sophisticated feature fusion. The variational models outperform the standard architectures in terms of average precision. With respect to the type of the model prior, the model that is integrated with the correlated Gaussian (FullARD) outperforms other variants, but at a small cost of pruning less weighted connections.

In Table 3, we added some experiments in order to high-

Table 1. Numerical results for object detection/segmentation trials on COCO (Lin et al., 2014). Average precision and precision on different threshold and object sizes are shown, alongside with network size and inference time (measured in milliseconds), for proposed models and other feature pyramid variants.

Network	Model	AP	AP_{50}	AP_{70}	AP_S	AP_M	AP_L	Params	Inference
Faster RCNN	BiFPN	0.293	0.486	0.311	0.163	0.316	0.350	1.60M	7.8 ± 0.01
	PANet	0.296	0.486	0.314	0.167	0.320	0.351	1.74M	6.7 ± 0.01
	NAS-FPN	0.307	0.509	0.326	0.175	0.342	**0.392**	1.53M	5.4 ± 0.10
	PConv	0.308	0.510	0.320	0.180	0.346	0.391	**1.25M**	8.4 ± 0.77
	HRNet	0.305	0.510	0.310	0.161	0.345	0.381	1.32M	$\mathbf{3.2 \pm 0.17}$
	ARD	0.315	0.525	0.331	**0.187**	0.340	0.377	1.67M	6.3 ± 0.01
	FullARD	**0.322**	**0.533**	**0.342**	0.186	**0.351**	0.388	1.74M	6.5 ± 0.01
Mask RCNN	BiFPN	0.271	0.451	0.284	0.109	0.291	0.402	1.60M	7.8 ± 0.01
	PANet	0.268	0.446	0.279	0.111	0.288	0.393	1.74M	6.7 ± 0.01
	NAS-FPN	0.280	0.468	0.290	0.117	0.308	0.411	1.53M	5.4 ± 0.10
	PConv	0.279	0.464	0.290	0.117	0.309	0.410	**1.25M**	8.4 ± 0.77
	HRNet	0.288	0.484	0.301	0.114	0.314	0.418	1.32M	$\mathbf{3.2 \pm 0.17}$
	ARD	0.290	0.481	0.303	0.124	0.315	0.424	1.67M	6.5 ± 0.01
	FullARD	**0.299**	**0.499**	**0.314**	**0.126**	**0.324**	**0.447**	1.74M	6.8 ± 0.02

Table 2. Numerical evaluation of uncertainty estimates for Faster RCNN trained on three different datasets. *Baseline* indicates detections acquired using the weight scaling rule and thresholded via the use of NMS, *Mean* detections are obtained with test time averaging and NMS applied and *Var voting* indicates predictions of time averaging but with the use of prediction variance coupled with the var voting algorithm. Ten forward passes where performed for each image.

Dataset	Model	AP	AP_{50}	AP_{70}
PlantDoc	Baseline	0.321	0.525	0.354
	Mean	0.333	0.533	0.364
	Var voting	**0.351**	**0.539**	**0.404**
COCO	Baseline	0.313	0.521	0.332
	Mean	0.286	0.451	0.315
	Var voting	**0.341**	**0.563**	**0.365**
Cards	Baseline	0.886	0.997	0.984
	Mean	0.889	0.999	0.989
	Var voting	**0.912**	**0.999**	**0.994**

light the benefits of pruning weights in a probabilistic manner. Specifically, we experimented with the initialized complex architecture of our model with no pruning; also, we randomly pruned a subset of weights and trained the resulting network via maximum likelihood (respectively "no pruning" and "random pruning" in Table 3). We tested the non-probabilistic method of Lasso pruning where a scaled regularization term based on the L_1 norm of the weights was added to the detector loss function. Finally, we added more sophisticated deterministic pruning methods. We experimented with the gradient-based pruning method in (Molchanov et al., 2019) and with the method of (Frankle & Carbin, 2019). Both methods require the fully connected network to be trained up to an optimal point, and then iterative connection pruning is applied. We set both methods to prune connections until 9 connections remain (in order to match ours and the Lasso-based pruning techniques).

We can see that the probabilistic methods of pruning yield better results than the deterministic ones. The correlated prior furthermore has the same accuracy as the fully complex model with no pruning while keeping only 25% of total connections. Additionally, both the variational and Lasso based methods yield better results than random pruning, verifying our notion that the network can learn to drop redundant connections. In Figure 2, we present two plots: the top plot depicts the active weights versus the training iteration, where all the methods progressively drop this number to a point where it reaches stability. In the bottom plot, we can observe some indicative values of the means of the approximate posterior on the weights.

Finally, we trained our proposed model with the ARD prior on the connection weights, integrated with the Faster RCNN network in three distinct datasets (COCO, Plants, Cards). By conducting these experiments we wanted to study the feature fusion architecture (i.e., corresponding to the non-pruned connection set after training). The datasets for this experiment where carefully chosen as each one bears its unique characteristics. Specifically, COCO (Lin et al., 2014) is an extremely demanding dataset containing multiple objects at different scales and sizes, PlantDoc (Singh et al., 2019) contains (mostly) medium and large objects and the Cards dataset (Crawshaw, 2020) is comprised solely of small objects. As we can see in Figure 3, each dataset leads to its own distinct optimal architecture. This means that the net-

Table 3. Numerical results for instance segmentation trials on COCO (Lin et al., 2014). Average precision (AP) is shown, alongside with network size (in terms of preserved connections, "Cons" and number of parameters, "Params") and inference time (measured in milliseconds) for different pruning schemes.

Model	AP	Cons	Inference	Params
No Pruning	0.299	63	14.2 ± 0.1	1.74
Random Pruning	0.222	16	8.1 ± 0.04	1.60
Lasso-based	0.283	9	$\mathbf{4.8 \pm 0.02}$	**1.32**
(Molchanov et al., 2019)	0.286	9	6.1 ± 0.03	1.38
(Frankle & Carbin, 2019)	0.280	9	7.1 ± 0.02	1.40
ARD	0.290	**9**	6.5 ± 0.01	1.39
FullARD	**0.299**	16	6.8 ± 0.02	1.60

work can learn to fuse and use those feature maps that are more valuable to each specific dataset.

4.2.1. Evaluating model uncertainty

We conducted experiments to quantify the uncertainty estimates of our model. By having the distribution of the connection weights w we can easily sample values of w. Each different sample from these distributions results to a distinct feature fusion network. Thus, instead of directly using the mean values of w at test time, we can first draw sample of weights acquiring a feature fusion network and then pass the image to it. This practice is referred to as test time averaging, and has been used for acquiring uncertainty estimates in stochastic neural networks (Kingma et al., 2015). Also, it has been previously applied in an objection detection architecture context (Miller et al., 2018).

We have performed 10 single forward passes (each time sampling different values of w), a process which yields a larger set $D = \{D_1, \ldots, D_{10}\}$ of 10 individual detection sets $D_i = B, S$ where B and S are sets containing the bounding boxes and the object scores respectively. Those prediction sets will have significant overlap, thus we sort the predictions according to their IoU and then we calculate the mean and variance of each prediction box resulting in $D_f = \{B_{mean}, B_{variance}, S_{mean}, S_{variance}\}$.

In order to numerically quantify those uncertainty estimates we used the variance voting ("var voting") algorithm proposed in (He et al., 2019), which modifies the non-maximum suppression (NMS) scheme. It uses the variance of a predicted location and refines each candidate bounding box location according to the learned variances of neighboring bounding boxes. The results are reported in Table 2. For all the datasets we can observe that the use of variance estimates can slightly improve network performance. We can observe the effectiveness of the var voting algorithm combined with our uncertainty estimates in Figure 4. The bounding boxes where refined to better match the target object. We can also observe model uncertainty for the predicted locations of bounding boxes, and we can see that accumulating predictions can even result in a prediction that a single forward pass could miss.

5. Conclusion and Future Work

We have presented Variational Feature Pyramid Networks, as extensions to the widely used FPN backbone. These networks are efficient and can be easily applied to various detectors for more efficient feature fusion. They can adapt to the underlying data, leading to specific fusion architecture for each training set. The Bayesian framework is used in our methods, allowing the user to capture uncertainty estimates about the trained model in contrast to deterministic variants. Numerical experiments show that the integration of the proposed model results in improving overall detection efficiency. For future work, we aim at exploring other forms of initial complex architecture with more hidden layers and modules. We also plan to experiment with different sparse prior distributions, allowing for more complex variational distributions over the model weights. As recent research does point out to non-Gaussianity (Fortuin et al., 2022), we plan to experiment with heavy-tailed alternatives [2]. We can envisage a Student's-*t* model using a Gaussian-Gamma factorization, and other options may include a Weibull or a Gamma distribution, in order to constrain weights as strictly positive following (Fan et al., 2020) or (Figurnov et al., 2018). We look forward to a fuller exploration of the pros and cons of all these models as future work.

Acknowledgements

We would like to thank the anonymous reviewers who gave us useful comments and helped improve this work. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan XP GPU used for this research. This research has been co-financed by the EU

[2] We have included a short discussion, derivation and numerical results for a model with a Laplace prior in the Appendix.

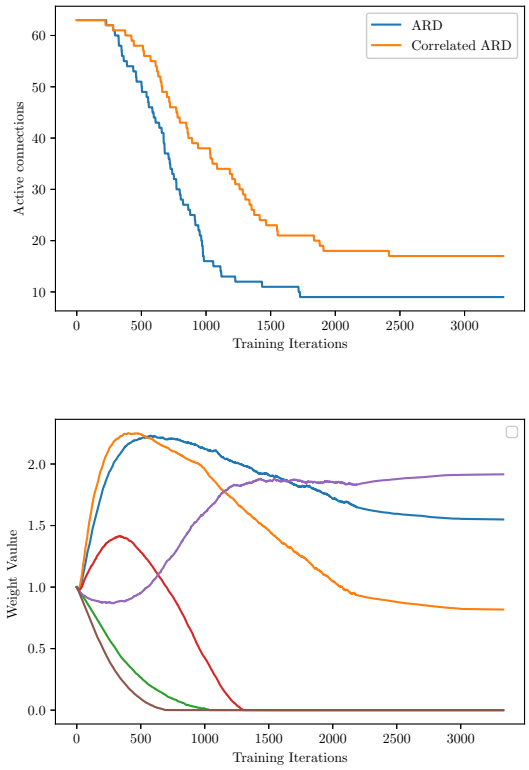

Figure 2. Top: Plot of the number of non-pruned weights/connections versus training iterations using different priors on the same setting (Faster RCNN on COCO). Bottom: Plot of indicative values of the means of the approximate posterior versus training iterations, over randomly picked network connections (each color corresponds to a different connection).

and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the calls "OPEN INNOVATION IN CULTURE" (project *Bessarion* - T6YBΠ-00214) and "RESEARCH - CREATE - INNOVATE" (project *Culdile* - code T1EΔK-03785); this research has also been co-financed by the Operational Program Epirus 2014-2020 (project EP1AB-0028216).

References

Bellido, I. and Fiesler, E. Do backpropagation trained neural networks have normal weight distributions? In *International Conference on Artificial Neural Networks*, pp. 772–775. Springer, 1993.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *Proceedings of the IEEE European Conference on Computer Vision*, pp. 213–229, 2020.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu,

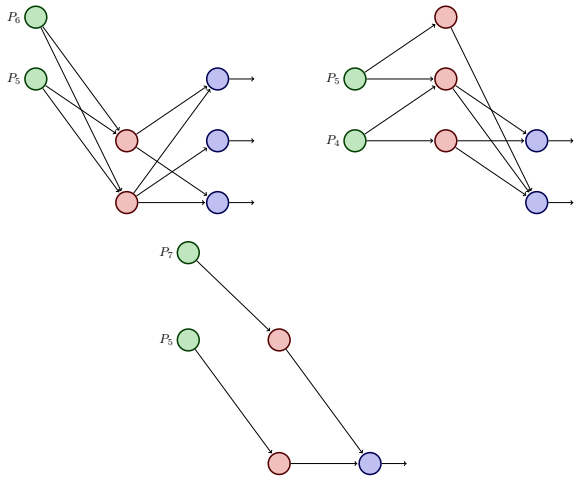

Figure 3. Plot of different resulting architectures for the trained model, combined with the proposed FullARD prior on Faster RCNN on three different datasets (top row: "COCO", "Plants", bottom row: "Cards").

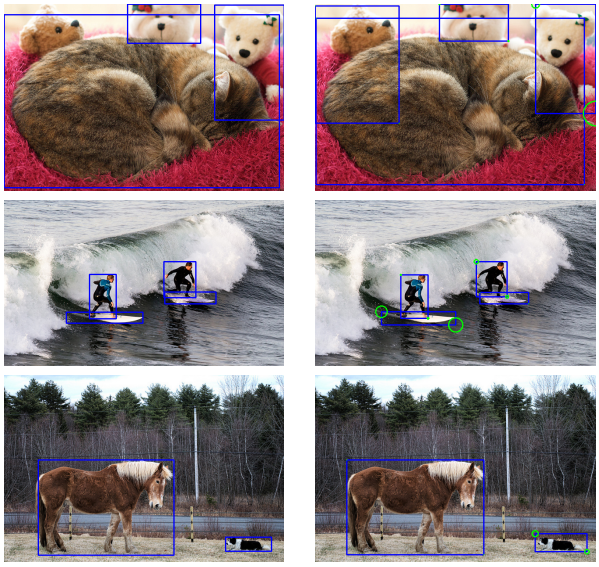

Figure 4. Results of standard NMS method (left column) and the Var voting results (right column) for the Faster RCNN network combined with the proposed method with ARD prior on COCO dataset. Bounding boxes are drawn with blue color, and the variance of each bounding box is outlined using green circles (variance is proportional to the radius around the respective bound point).

C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Chen, Q., Wang, Y., Yang, T., Zhang, X., Cheng, J., and Sun, J. You only look one-level feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13039–13048, 2021.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, 2017.

Crawshaw, A. Uno cards dataset. https://public.roboflow.com/object-detection/uno-cards, 2020.

Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., and Zhang, L. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7373–7382, 2021.

Fan, X., Zhang, S., Chen, B., and Zhou, M. Bayesian attention modules. *Advances in Neural Information Processing Systems*, 33:16362–16376, 2020.

Figurnov, M., Mohamed, S., and Mnih, A. Implicit reparameterization gradients. *Advances in Neural Information Processing Systems*, 31, 2018.

Fortuin, V., Garriga-Alonso, A., Ober, S. W., Wenzel, F., Ratsch, G., Turner, R. E., van der Wilk, M., and Aitchison, L. Bayesian neural network priors revisited. In *International Conference on Learning Representations (ICLR)*, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7036–7045, 2019.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch SGD: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

He, Y., Zhu, C., Wang, J., Savvides, M., and Zhang, X. Bounding box regression with uncertainty for accurate object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2888–2897, 2019.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, 28: 2575–2583, 2015.

Kong, T., Sun, F., Tan, C., Liu, H., and Huang, W. Deep feature pyramid reconfiguration for object detection. In *Proceedings of the IEEE European Conference on Computer Vision*, pp. 169–185, 2018.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 740–755, 2014.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017a.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017b.

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. Path aggregation network for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8759–8768, 2018.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Miller, D., Nicholson, L., Dayoub, F., and Sünderhauf, N. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3243–3249. IEEE, 2018.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pp. 2498–2507, 2017.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Petersen, K. B., Pedersen, M. S., et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: towards real-time object detection with Region Proposal Networks. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2016.

Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., and Batra, N. Plantdoc: A dataset for visual plant disease detection, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D., Mu, Y., Wang, X., Liu, W., and Wang, J. High-resolution representations for labeling pixels and regions. *arXiv preprint arXiv:1904.04514*, 2019.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10781–10790, 2020.

Tian, Z., Shen, C., Chen, H., and He, T. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9627–9636, 2019.

Tibshirani, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Tipping, M. E. The Relevance Vector Machine. In *Advances in neural information processing systems (NIPS)*, pp. 652–658, 1999.

Titsias, M. and Lázaro-Gredilla, M. Doubly stochastic variational bayes for non-conjugate inference. In *International Conference on Machine Learning*, pp. 1971–1979, 2014.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578, 2021.

Wang, X., Zhang, S., Yu, Z., Feng, L., and Zhang, W. Scale-equalizing pyramid convolution for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13359–13368, 2020.

Zhao, C., Ni, B., Zhang, J., Zhao, Q., Zhang, W., and Tian, Q. Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2780–2789, 2019a.

Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., and Ling, H. M2det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 9259–9266, 2019b.

A. Using a Laplace prior

A.1. Laplace Distribution

Our first option for the weight-connection prior had to be a Gaussian distribution, due to its simplicity, good analytical properties, and implications of the central limit theorem (Bellido & Fiesler, 1993). Recent research however does point out to non-Gaussianity of neural network weights (Fortuin et al., 2022), and we have indeed conducted (preliminary) experiments towards this direction, especially focusing on heavy-tailed alternatives. We experimented with the Laplace distribution which has heavier tails than the Gaussian and is discontinuous at $w = \mu$. It is often used in the context of Lasso regression, where it encourages sparsity in the learned weights (Tibshirani, 1996). Assuming that the weights are i.i.d., we set the prior distributions to zero-mean Laplace:

$$p(\mathbf{W}) = \prod_{i=1} p(\mathbf{w}_i) \;\; \text{where} \;\; \mathbf{w}_i \sim \text{Laplace}(0, 1). \tag{19}$$

The straightforward choice for the approximate variational distribution that satisfies the conditions for applying SGVB is the factorized Laplace distribution:

$$q(\mathbf{W}) = \prod_{i=1} q(\mathbf{w}_i) \;\; \text{where} \;\; \mathbf{w}_i \sim \text{Laplace}(\mu_i, \beta_i). \tag{20}$$

Thus the set we wish to optimize are the variational parameters $\phi = \{\mu, \beta_i\}$. We can easily draw samples from the variational posterior:

$$\mathbf{w} = \mu - \beta \text{sign}(\epsilon) \log(1 - 2|\epsilon| + \alpha) \;\; \text{where} \;\; \epsilon \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2}), \tag{21}$$

where the sign function evaluates the sign of ϵ and α is a parameter with small value introduce to provide numerical stability. The KL term of the VLB can be computed analytically as:

$$KL(q_\phi(\mathbf{W})||p(\mathbf{W})) = -\log(\beta) + |\mu| + \beta \exp(\frac{-|\mu|}{\beta}) - 1. \tag{22}$$

A.2. Experimental Results

Table 4. Numerical results for object detection/segmentation trials on COCO (Lin et al., 2014). Average precision and precision on different threshold and object sizes are shown, alongside with network size and inference time (measured in milliseconds), for proposed models and other feature pyramid variants.

Network	Model	AP	AP_{50}	AP_{70}	AP_S	AP_M	AP_L	Params	Inference
Faster RCNN	Laplace	0.312	0.517	0.328	0.179	0.346	0.392	1.602M	6.8 ± 0.10
Mask RCNN	Laplace	0.283	0.473	0.294	0.125	0.304	0.412	1.740M	6.9 ± 0.03

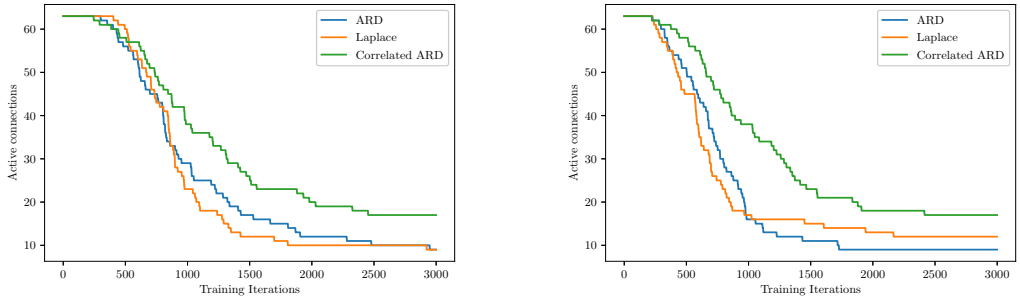

Figure 5. Plot of the number of non-pruned weights/connections versus the training iterations using different priors on the same setting on COCO, (left Mask-RCNN and right Faster-RCNN).