JMLR: Workshop and Conference Proceedings 63:334–349, 2016 ACML 2016

Bank of Weight Filters for Deep CNNs

Suresh Kirthi Kumaraswamy SURESH.KIRTHI@EE.IISC.ERNET.IN
P S Sastry SASTRY@EE.IISC.ERNET.IN
K R Ramakrishnan KRR2504@GMAIL.COM
Indian Institute of Science, Bengaluru, India.

Editors: Robert J. Durrant and Kee-Eung Kim

Abstract

Convolutional neural networks (CNNs) are seen to be extremely effective in many large object recognition tasks. One of the reasons for this is that they learn appropriate features also from the training data. The convolutional layers of a CNN have these feature generating filters whose weights are learnt. However, this entails learning millions of weights (across different layers) and hence learning times are very large even on the best available hardware. In some studies in transfer learning it has been observed that the network learnt on one task can be reused on another task (by some finetuning). In this context, this paper presents a systematic study of the exchangeability of weight filters of CNNs across different object recognition tasks. The paper proposes the concept of *bank of weight-filters* (BWF) which consists of all the weight vectors of filters learnt by different CNNs on different tasks. The BWF can be viewed at multiple levels of granularity such as network-level, layer-level and filter-level. Through extensive empirical investigations we show that one can efficiently learn CNNs for new tasks by randomly selecting from the bank of filters for initializing the convolutional layers of the new CNN. Our study is done at all the multiple levels of granularity mentioned above. Our results show that the concept of BWF proposed here would offer a very good strategy for initializing the filters while learning CNNs. We also show that the dependency among the filters and the layers of the CNN is not strict. One can choose any pre-trained filter instead of a fixed pre-trained net, as a whole, for initialization. This paper is a first step in the direction of creating and characterizing a *Universal BWF* for efficient learning of CNNs.

Keywords: CNN, deep learning, neural networks, transfer learning, bank of weigh filters, BWF

1. Introduction

Object recognition is an important problem in computer vision. Over the years many pattern recognition techniques (such as SVMs, random forests etc.) have been employed for it. In recent times deep neural networks or, more specifically, convolutional neural networks (CNNs) are seen to be phenomenally successful in many difficult object recognition tasks (Krizhevsky et al. (2012); Zeiler and Fergus (2013); Simonyan and Zisserman (2014); Szegedy et al. (2015)). An interesting aspect of CNNs (or, in general, of deep neural networks) is that feature generation part is fused with the classifier part and both parts are learned together using the training data.

In the more traditional pattern recognition techniques in computer vision, the features to be used are separately designed and classifier is then learnt using training data images represented using the chosen features. Thus feature generation part and classification part were mostly well separated and agnostic to each other. For example, a typical vision task may use SIFT(Lowe (2004)) and/or histogram of oriented gradients(HOG)(Dalal and Triggs (2005)) as prefixed features. On the other hand, the neural network viewpoint always emphasized the learning of appropriate features using the training data. (For example, an early paper that made backpropagation very popular emphasized the idea of learning appropriate internal representations (Rumelhart et al., 1988)).

One motivation for the neural networks view of the importance of learning appropriate features is an analogy with the human visual system where multiple layers of neurons seem to be measuring many different features which are all presumably brought together to evoke the object perception. In such a view, it is conceivable that many of the features that human visual system uses are universal in the sense that there are features which are useful for all image based pattern recognition tasks. Possibly such features are learnt through evolution of the species. Then an interesting question for computer vision is whether there exist such universally applicable features for computer vision systems and, if so, how do we identify them.

The idea of a *bank of weight filters* that we discuss in this paper is a small attempt at investigating this viewpoint empirically in the context of CNNs for object recognition.

The initial layers of a CNN are convolutional layers which consist of many filters. The later layers are the fully connected layers. In the usual view of CNNs, the convolutional layers are viewed as feature generators while fully connected (FC) layers are viewed as the classifier. In this context, *bank of weight filters (BWF)* could be a set of weight-filters (that is, a set of weight vectors characterizing the filters) collected from many different CNNs which are trained on different object recognition tasks. By building many CNNs trained on object recognition tasks we may be able to generate a corpus of weight filters that can then be easily reused for all other object recognition tasks. This immediately raises some interesting questions. Are the weight filters learnt in all convolutional layers universally useful or only those learnt in the early layers can be used for other image recognition tasks? Is the set of all filters in a single convolutional layer together form a useful bank of filters or can we create such a bank of filters by randomly choosing filters from different CNNs?

In this paper we present an empirical investigation of these questions. Our overall strategy is as follows. (We explain our experimental set-up in more detail later on). We choose a set of object recognition tasks and then train a standard CNN architecture on these tasks using the standard algorithms. We then create different new CNNs whose weight filters are set randomly by sampling from the weight filters of already trained CNNs. We do this at different levels of granularity (such as network-level, layer-level and filter-level). We then compare the accuracies achievable with the new CNNs on object recognition tasks when we train only the classifier part (that is, the fully connected layers). We also look at the accuracy and time trade-offs in finetuning the convolutional layers of CNNs initialized randomly from the bank of weight filters. Thus the investigations presented in this paper essentially address the issue of exchangeability of weight filters across CNNs which we feel is a very important issue. Our results indicate that the individual weight filters learnt by different CNNs are indeed exchangeable (in the sense that they are useful for other object recognition tasks) and thus support our view that this is an interesting direction to pursue for creating new CNNs for different applications. Having access to such pretrained weight filters can greatly reduce the training time (as well as, possibly, the training set size) for training CNNs on new object recognition tasks.

The idea of re-usability of weight filters has been investigated earlier in the framework of transfer learning. In many transfer learning tasks the pre-trained CNNs are finetuned with lower learning rate for convolutional layers and higher learning rates for FC layers as in (Karayev et al. (2013)). This is similar to our view of keeping the feature generation part fixed and classification part trained afresh. The works of (Yosinski et al. (2014); Agrawal et al. (2014)) extensively studied the re-usability and expressibility of convolutional layers. The material presented here is closer to (Yosinski et al. (2014)) who studied this in a structured way up-to layer-level granularity. Unlike that paper here we do more elaborate analysis which is more granular, starting from the network-level and going all the way till the filter-level. We also study the random choice of layers and filters. There have also been studies on the weight initializations and their impact on training time and quality as in (Krähenbühl et al. (2015)). Their first layer is training data dependent and the other layers are initialized from the k-means cluster centers of the previous layer activations. Here we initialize using weight-filters from the pretrained CNNs which is different from training-data-dependent initialization used in that paper. In the process we also report the time (iterations) taken to attain best accuracy which is faster by a factor of *2 to 5* compared to the normal-training of CNNs. To the best of our knowledge no prior work exists on the analysis of exchangeability of the filters to the level of granularity that is investigated here. We would like to reiterate that unlike (Krähenbühl et al. (2015)), our objective here is to propose, construct and show the efficacy of BWFs. Our motivation is to eventually work towards the idea of building *a universal set of weight filters* and characterizing them.

The rest of the paper is organized as follows. In section 2 we give a detailed description of our experiment design. Section 3 explains our choice of the CNN architecture and learning algorithm, the choice of different object detection tasks and also some implementation details. Section 4 describes the actual empirical studies done and presents the results. We conclude the paper in Section 5.

2. Bank of weight-filters and experimental design

The *Bank of weight-filters*(BWF) is defined as a set of weight filters from many pretrained CNNs. This set can be viewed at multiple levels. At the highest level, each element of BWF is the entire set of weights taken as a whole from an entire CNN. We can represent it as $BWF = \{W_{n_1}, W_{n_2}, ...\}$, where W_{n_i} is weights of *network-i*. At the lowest level, BWF would have individual weight-filter as its elements. This can be represented as $BWF = \{W_{f_1}, W_{f_2}, ...\}$, where W_{f_i} represents weights of a specific filter which may be in any one of the learnt CNNs. If a BWF is constructed using a large number of CNNs trained on a given set of tasks then such a BWF can serve as a *universal BWF* for other (related) tasks to initialize CNNs. This idea is illustrated in Fig. 1.

In Fig. 1, three levels of BWFs are illustrated. Fig. 1(a) shows network-level BWF where each element consists of the five convolutional layers of a network and the set consists of *n-elements* where n is the number of trained CNNs that we have. From this set an element can be randomly chosen and used to initialize a *target-net*. The *target-net* initialized by network-level BWF is called *target-net-NR* (NR for network random). Fig. 1(b) shows layer-level BWF whose elements consist of all weight filters taken together from a particular layer of a CNN. These elements would be collected from many pretrained CNNs. An element from this set can be chosen at random to initialize a layer of *taget-net* that would be called *target-net-LR.* (LR stands for layer-random). Along similar lines, we can have filter-level BWF as shown in 1(c). This is divided into as many subsets as the number of convolutional layers in the network. The elements of a subset would be weights filters of that particular layer from any of a number of pretrained CNNs. Now the *target-net FR* is a *filter random* network initialized by randomly chosen weights from the filter-level BWF. The details of construction of different target networks are explained further when we describe our empirical studies in section 4.

Next we describe our overall experimental design. All the CNNs we use have the same architecture as the Alexnet (Krizhevsky et al., 2012). We chose this because this is a very standard structure for CNNs. Thus all our networks would have five convolutional layers and three fully connected (FC) layers. There are 3 kinds of nets that we use here. First is *normal-CNN*, for which we do the standard random initialization of the weights and train all the layers of the network from the scratch. Second kind of network is the *noise-initialized-CNN*. For this we do the standard random initialization of all weights. But when training the network, we keep the convolutional layer weights fixed (at their initial values) and train only the fully connected layers, namely, layers *6, 7 and 8.* From now on, we refer to this type of training, where weights in convolutional layers are fixed at their initial values but the weights in the three FC layers are trained, as *3-FC* training. The third kind of net we use is called *target-net* whose convolutional layers are initialized by random sampling without replacement from different BWFs (that is, network-level, layer-level and filter-level BWFs). Our method of characterizing the performance of the BWFs and assessing the exchangeability of the weight-filters is as follows. We consider the *normal-CNN* as the networks that achieve maximum attainable accuracy on the task and the *noise-initialized-CNN* as the ones that represent the least attainable accuracy. We report the accuracy achieved by the respective *target-nets* with different BWF initializations to assess the efficacy of the idea of BWFs. Since our BWF initializations are only for the convolutional layers, in these target nets we use the 3-FC training. We show that the BWF initialized networks achieve much better accuracy than noise-initialized CNNs though the accuracy is some what less than that of *normal-CNN* for that task. We also use a CNN with the 5-convolutional layers and two of the three fully connected layers(layer 6 and 7) initialized from the pretrained CNNs and train only the final layer, namely, *layer-8*. We call this CNN as *FC-layer-8* trained. Such a CNN is seen to achieve better accuracy than *noise-initialized-CNN* but much worse than any BWF initialized CNNs. This *FC-layer-8* training was done just to show that convolutional layers are reusable but FC layers are not. This also provides justification for restricting BWFs in our study to the convolutional layers.

Apart from this we also did finetuning of the *target-net* CNNs by training the fully connected layers from random initializations and convolutional layers from BWF initializations. Finetuning was done to show that loss of accuracy due to BWF initialization is recoverable and also to show the advantage gained in terms of training time by BWF initialization. The parameter setting for *finetuning* is given in the *table-5* of appendix- A. We have not used differential learning rates for convolutional layers and fully connected layers as is popularly done in CNN finetuning. Wherever the CNNs were initialized by random sampling from the BWFs we experimented five times and training on *normal-CNNs* was done once. For this analysis we have trained about 200 CNNs totally, encompassing various initializations and training methods.

Before we present the results of our empirical study, we explain our choice of CNN architecture and different object recognition tasks along with the datasets used.

3. Choice of CNN and different object recognition tasks

3.1. Alexnet

As mentioned already we have used the same architecture as Alexnet for all our CNNs. This is because it is a standard structure and seen to be very effective in object recognition. For the sake of completeness we describe Alexnet briefly. (For more details refer (Krizhevsky et al., 2012)). Alexnet is a CNN with five convolutional layers and three fully connected layers followed by a softmax layer for classification. The convolutional layers are incorporated with fixed dimensional kernel and stride for each layer. The stride and kernel width is progressively reduced across layers. The number and size of kernels of five conv layers are: 96 each of size $11 \times 11 \times 3$, 256 each of size $5 \times 5 \times 48$, 384 of size $3 \times 3 \times 256$, 384 of size $3 \times 3 \times 192$ and 256 of size $3 \times 3 \times 192$. Each of the three FC layers have 4096 neurons and the last is a softmax max layer with 4096 inputs and 1000 outputs. Unlike the traditional neural nets, the Alexnet uses the Rectified Linear Unit(ReLU) as the non-linearity which is found to reduce training time. Normalization in Alexnet is done across small neighborhood of responses from adjacent kernels, and not the entire layer. Alexnet uses *max-pooling layers* after each convolutional layer.

In our work we use *Caffenet* (Jia et al. (2014)) implementation of Alexnet. The parameter settings used are provided in the appendix- A. For training *normal-CNN*, we use parameters specified in *table-4* in appendix- A. The parameter setting for *finetuning* is given in the *table-5* of appendix- A.

3.2. Object recognition tasks

In order to explain the effectiveness of BWFs and exchangeability of filters in the CNNs, we would need to define different kind of tasks which are related. We create these different tasks as follows. Starting with ISLVRC12 (Russakovsky et al. (2015)) we choose five-subsets each consisting of 10 classes. The five subsets chosen are named set-1 to set-5 and their ILSVRC12 class numbers are 1-10, 991-1000, 501-510, 511-520 and 10 randomly chosen ones, respectively. The classes in the subsets belong to both man-made/natural categories. These five subsets can be thought of as five different tasks of object recognition. Table.1 lists the class names of each subset(one per class). Fig. 2 shows 64 sample images in each subset. One can notice the diversity of the classes, the color and the background in the images used.

For the sake of completeness we briefly describe the dataset from which our object recognition tasks are formed. The ILSVRC2012 (Russakovsky et al. (2015)) dataset consists of 1000 classes with 1.2million training images, 50,000 validation images and 100,000 testing images, among which majority are color and full resolution (as captured by the camera).

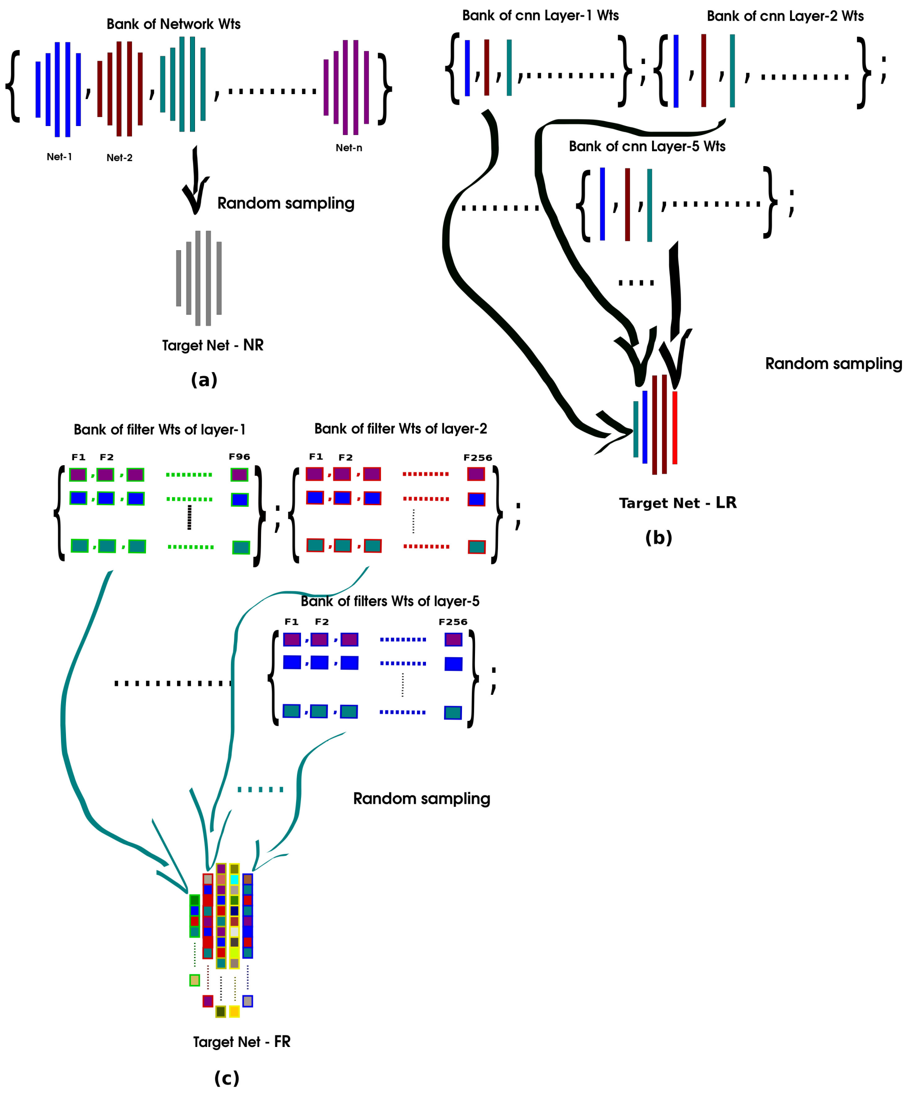

Figure 1: Illustration of bank of weight filters and random sampling of weights from BWF for weight initialization of *taget-nets*. (a) shows *Network-level-BWF*, a set with each element being a pretrained network, (b) shows *Layer-level-BWF*, with five subsets corresponding to five layers of pretrained CNNs and (c) is *Filter-level-BWF* with five subsets corresponding to five layers and elements of each subset is a weight-filter, collected from many pretrained CNNs. Target-net-NR is initialized by randomly choosing an element from a network-level-BWF. Target-net-LR is a layer-random target-net, which is initialized by randomly choosing a layer from each subset of layer-level-BWF. Target-net-FR is a filter-random target-net, where each layer of a target-net is initialized with a sufficient number of weight-filters randomly chosen from a subset corresponding to that layer using the *Filter-level-BWF*.(*This figure is best viewed in color*)

The images were collected by web searching with the synonym sets (synsets) of words representing 1000 classes. The synsets of 1000 classes are non-overlapping. The images are manually annotated and verified for 99.7% precision across synsets. The classes consists of natural objects like birds, animals, fruits, beach, mountains, lake side and man-made objects like tables, chair, car, musical instruments and many more. Each class has about 1300 images and most of them are color images.

Class Names in each subset				
set-1	**set-2**	**set-3**	**set-4**	**set-5**
tench	buckeye	cliff dwelling	container ship	drake
goldfish	coral fungus	cloak	convertible	isopod
white-shark	agaric	clog	corkscrew	hyena
tiger-shark	gyromitra	cocktail shaker	cornet	electric guitar
hammerhead	stinkhorn	coffee mug	cowboy boot	loudspeaker
electric ray	earthstar	coffeepot	cowboy hat	speedboat
stingray	Grifola frondosa	coil	cradle	window shade
cock	bolete	combination-lock	crane	wreck
hen	ear	keyboard	crash-helmet	pomegranate
ostrich	toilet-tissue	confectionery	crate	potpie

Table 1: Ten class names with one name per class in each of the 5-subsets used.

4. Experiments with different levels of BWFs

In the next few subsections we present results of our empirical studies on each level of BWF.

4.1. Network level BWF

A network level BWF is a set, each of whose elements consists of weights of all the convolutional layers of a network, taken as a whole, from pre-trained CNNs. We construct this set by training *normal-CNNs* on each of the five 10-class object recognition tasks described in section 3.2. These object recognition tasks/datasets would be referred to as *set*-**1** to *set*-**5** This way we would have the weights from five-CNNs trained separately on the five tasks and this serves as our *five-element* network level BWF,

$$\mathrm{BWF}_{net\text{-}wts} = \{W_{cnn_{set\mathbf{1}}}, W_{cnn_{set\mathbf{2}}}, ..., W_{cnn_{set\mathbf{5}}}\}, \tag{1}$$

where $W_{cnn_{set\mathbf{i}}}$ consists of the weights of all convolutional layers of a CNN trained on dataset *set*-**i**.

These normal CNNs are trained with learning rate, $lr = 0.005$, reduced to one-tenth after 100,000 iterations and for a total of 150,000 iterations (refer Appendix- A). The target-nets for each task *set-i* was initialized with the weights of a net randomly chosen from the nets trained on datasets *set-j*, where $i, j \in \{1, 2, ..., 5\}, i \neq j$. We re-trained the target-net in two ways. First, we trained only the *FC-layer-8* with standard initializations of *FC-layer-8*.

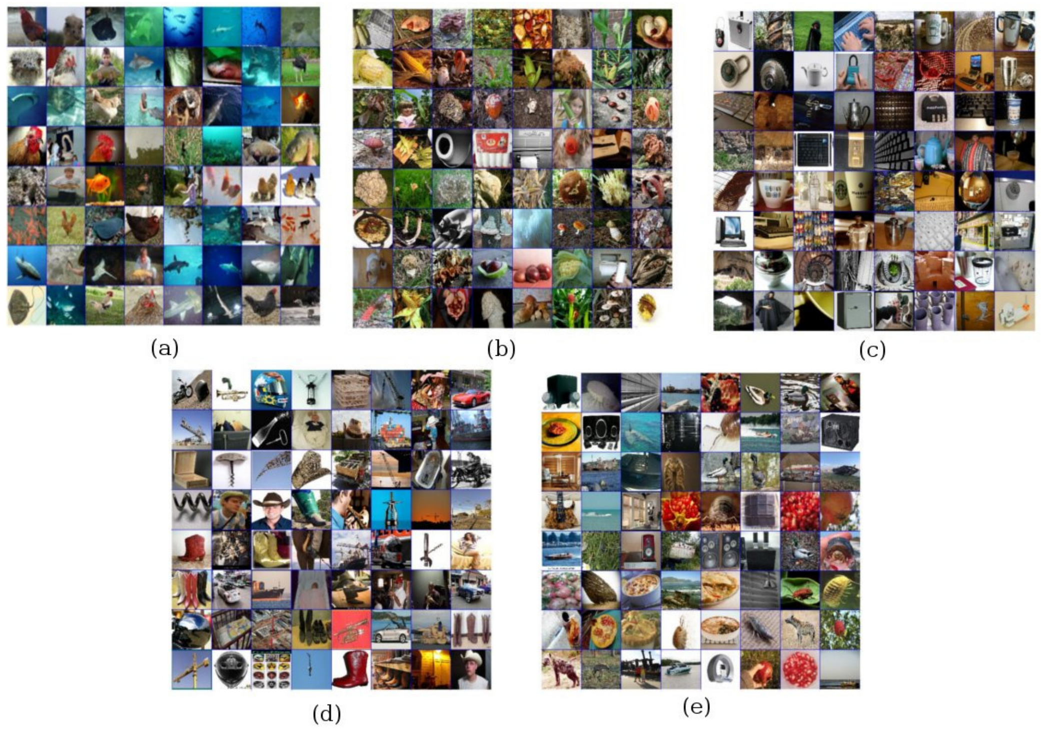

Figure 2: Randomly chosen representative images in each of the five subsets.

Second, we trained only the 3-FC layers with standard initialization of the FC layers. (See appendix. A for details of initializations).

These results are presented in Table 2. The top row in the table indicates the different training strategies used for the results presented in different columns of the table. These consist of normal-CNN , 3-FC training and FC-layer-8 training. The second row of the table indicates the initialization method used for the results in different columns. For normal-CNN and FC-layer-8 training the initialization is as in appendix- A. For the 3-FC training there are many possible initializations: noise-init, network-level BWF (denoted by NR), layer-level BFW (LR) and filter-level BWF (FR). In this subsection only results under NR are discussed. The LR and FR results are discussed in the next two subsections.

We found that *FC-layer-8* retrained target-net had accuracy which was better than that of the *noise-initialized-CNNs* but lesser than that of the target-net-NR with 3-FC layers retrained, as can be seen from Table. 2. The target-net with the 3-FC layers retrained had accuracy in the range of 60-70% which is much higher than that of *noise-initialized-CNNs* which was about 25%. This implies that fixing the weights of convolutional layers using the corresponding weights from other pretrained CNNs (which are trained on different image recognition tasks) will still give us fairly good performance. The performance of *FC-layer-8* trained target-net was inferior to the net with 3-FC training, strengthening the popular notion that in CNNs, the convolutional layers are more reusable than the FC layers.

Nature of re-training in most of the transfer learning tasks using CNNs, like in the case of imagenet to PASCAL dataset adaptation (Oquab et al.), flickr finetuning (Karayev et al., 2013) and others, is by initialization of the entire convolutional layers or initialization till *fully connected layer-7*. In many such adaptation/transfer-learning applications the configuration of the CNNs, the weight arrangements, are taken as sacrosanct and customization is confined to FC and softmax layers only. This is like our network-level BWF.

The network-level BWF we considered is the *target-net-NR* as shown in Fig. 1 (a). We take the performance achieved by the network-level BWF as a baseline and pursue the question of dependency of the filters to their position in the CNN network architecture. This is in keeping with our motivation of exploring exchangeability of weight filters at different levels of granularity.

4.2. Layer level BWF

We define $layer\text{-}level BWFs$ as a set whose elements are layer weights of trained CNNs. The $layer\text{-}level\text{-}BWF$ we use here can be specified as follows,

$$\begin{aligned} BWF_{layer\text{-}wts} = \{ & W^{\mathbf{L}_1}_{cnn_{set\mathbf{1}}}, W^{\mathbf{L}_1}_{cnn_{set\mathbf{2}}}, \ldots W^{\mathbf{L}_1}_{cnn_{set\mathbf{k}}}; \\ & W^{\mathbf{L}_2}_{cnn_{set\mathbf{1}}}, W^{\mathbf{L}_2}_{cnn_{set\mathbf{2}}} \ldots., W^{\mathbf{L}_2}_{cnn_{set\mathbf{k}}}; \ldots\ldots\ldots\ldots \\ & W^{\mathbf{L}_m}_{cnn_{set\mathbf{1}}}, W^{\mathbf{L}_m}_{cnn_{set\mathbf{2}}}, \ldots., W^{\mathbf{L}_m}_{cnn_{set\mathbf{k}}}; \}, \end{aligned} \tag{2}$$

where $W^{\mathbf{L}_j}_{cnn_{set\mathbf{i}}}$ consists of all the weights of j^{th} convolutional layer of the CNN trained on dataset $set{-}\mathbf{i}$.

A $layer\text{-}levelBWF$ would treat the weights of all the filters in a convolutional layer as a single peice as far as exchangeability is concerned. We can think of it as having subgroups with each subgroup consisting of a particular convolutional layer weights from all pretrained

CNNs. In this way the Alexnet would have five sub-groups in the *layer-level$BWFs$* corresponding to each convolutional layer. We use the layer-level BWF for creating a target-net by randomly choosing each convolutional layer of the this network from each sub-group of the *layer level BWF* as shown in the Fig. 1(b). The target-net built like this could be having each layer picked from CNNs trained on different datasets for different tasks. We call this as *target-net-LR* where LR stands for *layer-random*. Suppose $W^{\mathbf{L}_j}_{cnn_{set\mathbf{r}}}$ represents the weights in the j^{th} convolutional layer of the target-net for dataset $set{-}\mathbf{r}$. While creating the target-net, we make sure to use only $W^{\mathbf{L}_j}_{cnn_{set\mathbf{s}}}$ from $BWF_{layer-level}$, such that $r \neq s$.

After initializing target-net-LR using randomly chosen layers from BWF$_{layer\text{-}wts}$, we train only the three FC layers. The accuracy of the target-net-LR, thus trained, is in the range of 52 to 69% as can be seen from the results presented in Table 2. This is about 10% lower than the *target-net-NR*, but much higher than the FC-trained random initializations or FC-layer-8 re-trained net. Recall that in target-net-LR, weights of different convolutional layers are fixed from different pretrained CNNs (and these weights are not modified for the resullts discussed here). Surprisingly the CNNs are quite robust to the alteration of the net configuration. Equipped with the robustness evidence of CNNs to the configuration change we next go an extra step and see what happens if we create the target-net by random choice of individual filters from pretrained CNNs.

Train.	Normal	3-FC	FC-layer-8	3-FC		
Init.	*App A.*	noise-init	7-layers	NR	LR	FR
set-1	0.716	0.209 ± 0.07	0.467 ± 0.05	0.681 ± 0.03	0.584 ± 0.08	0.586 ± 0.08
set-2	0.810	0.282 ± 0.06	0.492 ± 0.07	0.720 ± 0.04	0.626 ± 0.09	0.625 ± 0.09
set-3	0.732	0.274 ± 0.04	0.379 ± 0.06	0.609 ± 0.06	0.528 ± 0.03	0.545 ± 0.05
set-4	0.774	0.214 ± 0.04	0.432 ± 0.11	0.670 ± 0.06	0.595 ± 0.08	0.605 ± 0.08
set-5	0.864	0.228 ± 0.06	0.562 ± 0.11	0.782 ± 0.04	0.695 ± 0.09	0.687 ± 0.09

Table 2: Accuracy of the nets with normal, noise and 7-layers (from pre-trained CNN) initialization along with the target-nets with *Network-Random(NR)*, *Layer-Random(LR)* and *Filter-Random(FR)* initialization. The training done here are normal, 3-FC and FC-layer-8.

4.3. Filter level BWF

We define *filter-level$BWFs$* as a set whose elements are individual filter weights of trained CNNs. A *filter-levelBWF* would be organized into subgroups with each subgroup containing filters from a particular convolutional layer from all pretrained CNNs. Thus, the Alexnet would have five sub-groups in the *filter-level$BWFs$* corresponding to the five convolutional layers (The different subgroups for *filter-levelBWF* exists due to the variation in filter sizes across convolutional layers). The *filter-level-BWF* we use here is specified as follows,

$$\begin{aligned} BWF_{filter\text{-}wts} = \{ & W^{\mathbf{F}_1}_{L_{1cnn_{set\mathbf{1}}}}, W^{\mathbf{F}_2}_{L_{1cnn_{set\mathbf{1}}}}, ...W^{\mathbf{F}_{k_{L1}}}_{L_{1cnn_{set\mathbf{1}}}}; \\ & W^{\mathbf{F}_1}_{L_{2cnn_{set\mathbf{1}}}}, W^{\mathbf{F}_2}_{L_{2cnn_{set\mathbf{1}}}}, ...W^{\mathbf{F}_{k_{L2}}}_{L_{2cnn_{set\mathbf{1}}}}; .... \\ & W^{\mathbf{F}_1}_{L_{5cnn_{set\mathbf{5}}}}, W^{\mathbf{F}_2}_{L_{5cnn_{set\mathbf{5}}}}, ..., W^{\mathbf{F}_{k_{Lj}}}_{L_{j_{cnn_{set\mathbf{5}}}}}; \}, \end{aligned} \tag{3}$$

where $W^{\mathbf{F}_k}_{L_{j_{cnn_{set\mathbf{i}}}}}$ is the weight vector of *filter-k* of *layer-j* of the CNN trained on *set-$\boldsymbol{i}$* and k_{Lj} is the total number of filters in *layer-j*.

We now create our target-net by randomly choosing each filter of each layer of the network from the appropriate sub-group of the $filter\text{-}levelBWFs$ as shown in the Fig. 1(c). The target-net built like this could be having each weight-filter picked from CNNs trained on different datasets for different tasks. We call this as *target-net-FR* where FR stands for *filter-random*.

In our target-net, let $\bar{W}^{\mathbf{F}_v}_{L_{j_{cnn_{set\mathbf{r}}}}}$ denote the weights for the *filter-v* of *layer-j* of the target-net meant for dataset *set*-$\mathbf{r}$. These weights are chosen by randomly sampling from $W^{\mathbf{F}_u}_{L_{j_{cnn_{set\mathbf{s}}}}}$ where u can be anything and $r \neq s$. After forming the target-net-FR like this by randomly choosing filters, we train the three FC layers. As can be seen from the results presented in Table 2, the accuracy of *the target-net-FR*, thus trained, is in the range of 54 to 68.7%. This is about 10% lower than the *target-net-NR*, but much higher than the FC-trained random initializations or FC-layer-8 re-trained net. What is very interesting is that the performance of *the target-net-FR* is on par with that of *the target-net-LR*, showing that the CNNs are very robust to the alteration of the net configuration. This seems to suggest that the interdependence among filters due to their co-learning is very minimal (or non-existant) at the layer level. This supports our view of a possible universal BWF with reusable weight filters. The BWF at filter level also provides more flexibility than the one at network level.

So far, in all the target-nets, the weights in the convolutional layers are fixed from BFWs (formed using other pretrained CNNs). While the performance is good, there is still some drop in performance compared to a normally trained CNN which is trained for the task at hand. We next address this issue by finetuning target-nets.

4.4. Finetuning

By finetuning we mean a training process where the weights in the convolutional layers are initialized through appropriate BWFs and then these weights are also adjusted along with the weights of FC layers (which are initialized randomly) in the normal training process. This is unlike *FC-3* training where we trained only the fully connected layers. The results of finetuning using different initializations and on the different datasets are shown in Table. 3. In the table, the Normal-CNN refers to initializing all weights randomly and then training all weights. The specific learning parameters are given in Appendix. The NR, LR and FR in the table refer to initializing the convolutional layer weights using the different BWFs.

As can be seen from the table, post-finetuning, all the initializations using *network level, layer level and filter level BWF*, achieve almost the same accuracy as that of the CNNs

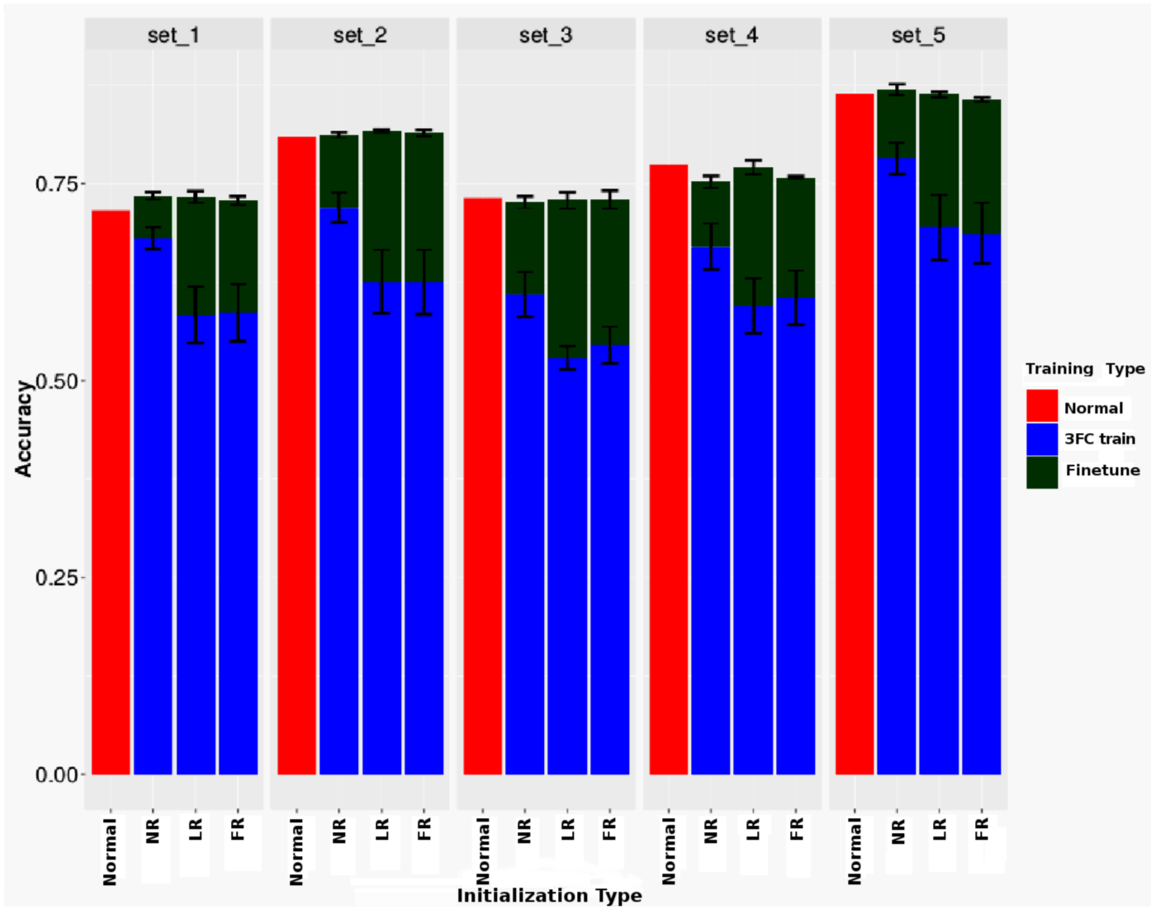

Figure 3: Plot of Accuracy versus initialization type (showing standard deviation for 5-trials), with *normal, NR, LR and FR* type of initializations and *normal, 3-FC layers and finetuning* type of trainings. The accuracy of all the initializations are similar once finetuned. *Note*: The bars of *3-FC layers and finetuning* are overlapping with only the gain due to finetuning is visible in *green* and the normal CNN accuracy is for a single trial on each set. (*This figure is best viewed in color*)

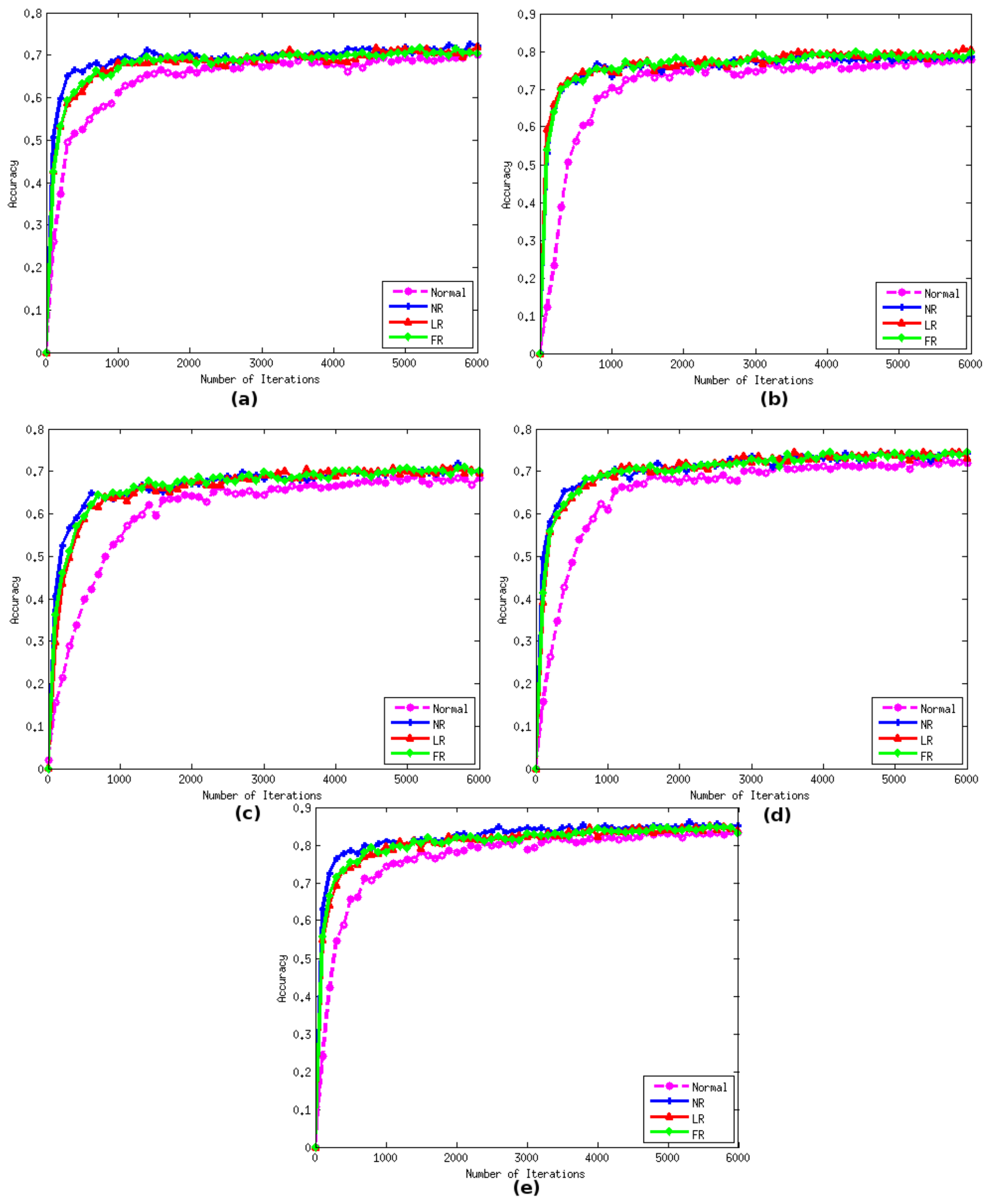

Figure 4: Accuracy versus Number of iterations (mean of 5-trials) for normal-CNN, and NR, LR and FR type of initializations. The subplots *a-e* refer to set-1 to set-5 respectively. Notice that the number of iterations required by the normally initialized CNN to reach its best performance is much more than that by NR, LR and FR initialized CNN. *(This figure is best viewed in color)*

trained normally.[1] Thus after initializing the weight filters using our BWF, by finetuning we can get the same performance as by learning a CNN from scratch. Fig. 3 shows the accuracies gained by finetuning on different initializations. The target-nets NR, LR and FR gained 10%, 20% & 20% respectively, as seen from Fig. 3. The finetuning takes fewer iterations to learn because the target-nets NR, LR and FR already have filter weights that are quite good. The number of iterations taken by normally trained CNN in comparison to those for NR, LR and FR initialized and finetuned CNNs to reach a stable performance are shown in Fig. 4. As can be seen the NR, LR or FR initialized CNNs needed much fewer iterations. These took about 300-500 iterations while normal-CNN took 1500-2500 iterations. This brings out the efficiency gained by using BWFs for initializing the weights in the convolutional layers.

Training	Normal-CNN	Finetune		
Init.	*ref:App. A.*	NR	LR	FR
Set-1	0.716	0.734 ± 0.011	0.733 ± 0.017	0.728 ± 0.013
Set-2	0.810	0.812 ± 0.008	0.817± 0.004	0.814 ± 0.009
Set-3	0.732	0.726 ± 0.017	0.729± 0.023	0.730 ± 0.025
Set-4	0.774	0.752 ± 0.017	0.771± 0.019	0.758 ± 0.005
Set-5	0.864	0.869 ± 0.015	0.863± 0.008	0.856 ± 0.007

Table 3: Accuracies of the target-nets (along with standard deviation for 5-trials), with *Network-Random(NR), Layer-Random(LR) and Filter-Random(FR)* initializations after finetuning. Here *normal-CNN* refers to network trained normally from random initialization of all weights. (The parameters for learning as in Appendix-A). As we can see, the accuracies of the target-nets NR, LR and FR are similar to that of *normal-CNN* for all the datasets.

5. Conclusion

CNNs are highly successful for many computer vision tasks such as *object recognition, segmentation etc.* They are also seen to be very effective in areas such as speech and NLP. However, training a CNN needs a large number of examples as well as large computational effort. The CNNs essentially learn to extract useful features by the process of adapting the filter weights in the convolutional layers. Hence, it is reasonable to suppose that many of the filters learnt by any CNN in an image-based pattern recognition task should be useful in other such task too. In this paper we presented an empirical investigation of this issue of exchangeability of weight filters of CNNs at different levels of granularity. We proposed the idea of a *bank of weight filters* which is a repository of filters from other trained CNNs as a means to initialize convolutional layers of a target CNN. We showed that even random choice of filters from other nets gives us fairly good accuracy. We also showed that by

1. All the proposed initializations for set1 seem to outperform normal training consistently. The reason could be that most of set1 images have sea/ocean ambiance and are uncluttered. But we did not investigate this specifically because the motivation here is to only explore the applicability of the idea of BWFs.

finetuning a net initialized like this, we can get the same accuracy as a CNN trained from scratch but takes much fewer iterations for training. Further our results also show that the individual filters learnt in a CNN are fairly exchangeable.

We feel the results presented here provide enough justification for our view that creating a universal BWF may be a good way to make CNN learning more efficient. In this paper we have treated all convolutional layers as same. However, it is possible that the filters learnt in early layers may be more universal (in the sense that they are useful in many other tasks) as compared to filters learnt in later layers. In this paper we explored only random choice from the BWFs. In general, to initialize a new network we should take a set of filters which have high level of 'diversity' and low level of 'redundancy' among them. Thus, better characterization of BWFs with better strategies for sampling from them to initialize target-nets may be needed for efficient learning of CNNs. All such issues are important in characterizing a universal BWF and we would be exploring some of these issues in our future work.

References

Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *Computer Vision–ECCV 2014*, pages 329–344. Springer, 2014.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. Recognizing image style. *arXiv preprint arXiv:1311.3715*, 2013.

Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Weakly supervised object recognition with convolutional neural networks.

DE Rumelhart, GE Hinton, and RJ Williams. Learning internal representations by error propagation. In *Neurocomputing: foundations of research*, pages 673–695. MIT Press, 1988.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

Appendix A. Parameters used for training Alexnet

These are the standard parameters used in caffenet implementation of Alexnet (Jia et al. (2014)). *Table-4* contains the parameters used for *normal-CNN* and all the *target-nets*, network random, layer random and filter random. In *Table-5* we have given the parameters used for finetuning.

Normal training								
Test		Learning-rate						
Iters	Interval	Base	Policy	Gamma	Step-size	Momentum	Wt-decay	Iters
1000	1000	0.005	*step*	0.1	100K	0.9	0.0005	150K
Finetuning								
100	100	0.005	*step*	0.1	10K	0.9	0.0005	40K

Table 4: Parameters of Caffe implementation of Alexnet.

Initialization - (with *Gaussian* $\sim (\mu, \sigma)$ where $\mu = mean, \sigma = standard\ deviation$).							
CONV-1	CONV-2	CONV-3	CONV-4	CONV-5	FC-6	FC-7	FC-layer-8
$0, 0.01$	$0, 0.01$	$0, 0.01$	$0, 0.01$	$0, 0.01$	$0, 0.005$	$0, 0.005$	$0, 0.005$

Table 5: Initialization of layers.