Forward Learning Convolutional Neural Network

Hong Hu, Xin Hong, Dan Yang Hou, Zhongzhi Shi

Forward Learning Convolutional Neural Network

Hong Hu[1], Xin Hong[1,2], Dan Yang Hou[1,2], and Zhongzhi Shi[1]

[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{huhong,shizz}@ict.ac.cn
icshongxin@163.com,littlemk@qq.com

Abstract. A conventional convolutional neural network (CNN) is trained by back-propagation(BP) from output layer to input layer through the entire network. In this paper, we propose a novel training approach such that CNN can be trained in forward way unit by unit. For example, we separate a CNN network with three convolutional layers into three units. Each unit contains one convolutional layer and will be trained one by one in sequence. Experiments shows that training can be restricted in local unit and processed one by one from input to output. In most cases, our novel feed forward approach has equal or better performance compared to the traditional approach. In the worst case, our novel feed forward approach is inferior to the traditional approach less than 5% accuracy. Our training approach also obtains benefits from transfer learning by setting different targets for middle units. As the full network back propagation is unnecessary, BP learning becomes more efficiently and least square method can be applied to speed learning. Our novel approach gives out a new focus on training methods of convolutional neural network.

Keywords: Forward Learning, Convolutional Neural Network, Transfer Learning, Extreme Learning Machine

1 Introduction

A convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery [13][1]. A CNN consists of an input and an output layer, as well as multiple hidden layers[13]. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, and normalization layers which play the role as feature extractor. An fully connected layer is applied at the top of feature extractor to classify extracted features. Convolutional layers apply a convolution operation to layer input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli [6].

Deep learning discovers intricate structure in large data sets by using the back-propagation algorithm proposed by Hinton in 1986. [16][17][2][7].

Deep convolutional nets have great breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech[5][18][4][3].

Although, the traditional BP has approved its ability to train the deep neural networks, the necessary of whole path feeding back from the output layer to the input layer at every cycle training limits the possibility of personalized learning of each units. And fast learning approaches such as ELM can't be applied in training every units of full deep neural networks. This gives rise to a high time cost. In fact, the weights of a CNN layer only pay attention to the output of the layer before, the efficient of a CNN layer's weights lies on the ability to correctly classify the target, the whole path feeding back of BP is unnecessary in most times. A deep network can

be divided into several units, and every unit contains several layers of convolution and pooling. We refer these units to forward unit. We stack these units and proposed a novel feed forward training approach. In our training approach, we train units one by one from input to output.

By adding auxiliary classifiers connected to these intermediate units, we would expect to encourage discrimination in the lower stages of the network, increase the gradient signal that gets propagated back, and provide additional regularization. [19]

During training process, in order to make every forward unit responsible to the classification, for every forward unit there are temporary fully connected layers applied to train the convolutional kernels in this unit. The training of a unit is typically training of a shallow ConvNet. In this way, the training becomes very simple and fast. Some fast learning approaches e.g. extreme learning machine (ELM) can be applied in this model. Our novel approach is denoted as forward learning convolutional neural network(FLCNN).

It is the first time that feed forward learning introduced into deep ConvNet. The main contributions of our work can be summarized as follows:

- In most cases our novel feed forward approach has similar performance with the traditional approach than perform BP through full network.The feed forward learning can be done one unit by unit, so least square approach e.g. extreme learning machines(ELM), can be applied into learning, such kind forward learning saves much time than back propagation over whole network.
- Different targets can be applied to training forward units, so such kind approach has the same benefits with transfer learning.
- The feed forward learning adds units one by one, so we can select suitable coefficients in units one by one, it is easy to find the suitable coefficients of layers.

2 The Principle of Forward Learning Convolutional Neural Network

The convolution pyramids or hierarchical convolutional factor analysis proposed by Kunihiko Fukushima in the 1980s in the deep learning is just a simulation of the columnar organization of our brains' primary visual cortex. Many functions of the primary visual cortex are still unknown, but the columnar organization is well understood[14] . The lateral geniculate nucleus (LGN) transfers information from eyes to brain stem and primary visual cortex (V1) [14] .

Columnar organization of V1 plays an important role in the processing of visual information. [12]. The principle of the convolution pyramids or hierarchical convolutional factor analysis is based on the following mathematical facts:

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learn-able filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter and the input which produces a 2-dimensional activation map of that filter. As a result, the network learns filters will activate when it detects some specific type of feature at some spatial position of the input. These specific types of features are just the local textures of an image.

The convolution kernel can be viewed as a kind of template. As we know, the content of an image is determined by the local textures of this image, and local textures are defined by image small blocks in a series small windows. During the training of every unit, the best local features are selected by local classification through an temporary fully connection layer.

ReLU is the abbreviation of Rectified Linear Units. It increases the nonlinear properties of the decision function without affecting the receptive fields of the convolution layer. After mapping with ReLU, the convolution results can be viewed as some kind of fuzzy values matching by logical

templates. The mapping functions is a non-saturating activation function $f(x) = \max(0, x)$ $f(x) = \max(0, x)$. Other functions are also used to increase non-linearity, such as the saturating hyperbolic tangent $f(x) = \tanh(x)$, and the sigmoid function $f(x) = (1 + e^{-x})^{-1}$. ReLU is preferable to other functions, because it trains the neural network several times faster[11] without a significant penalty to generalization accuracy.

Pooling layer, which is a form of non-linear down-sampling is responsible for determining which template a small image block belongs to. There are several strategies to implement pooling among which max pooling is most common used. The max pooling tries to find the most suitable matching position of a template. The pooling layer operates independently on every slice and reduce the input spatially. The most common form is a pooling layer with filters of size 2×2 applied with a stride of 2 down samples at every depth slice in the input by 2 along both width and height, discarding 75% of layer input. In this case, every max operation tries to find the best matching over 4 numbers. The depth dimension remains unchanged.

In [9], a CNN structure is summarized as some kind of granular computing. As a granular computing, template matching and histogram statistics are used alternatively, the focuses of CNNs are enlarged along the way from input to output. As we know, template matching is sensitive to image transformation, e.g. shift, rotation, scaling and so on. At other hand, histogram only counts the frequency of templates distribution over an image. Features abstracted by histogram is more robust than template matching. In a histogram, the locations of templates are neglected. A histogram, which is a vector, can be easily computed by a special full connected layer.

If every histogram vector of images in training set has enough information about the content of images, the classification of this image set can be completed by Support vector machine(SVM) over their histogram vectors of images or a fully connected layer. Otherwise, some important location information of local textures is missing in these histogram vectors while larger templates should be used to recognize more detail about images.

In most cases, if a fully connected neural layer is applied after this convolution layer, and a high precision of classification is achieved, larger templates are unnecessary, otherwise one more convolution layers is needed. So in most cases, ConvNets can be trained layer by layer or several layers by several layers from input to output. Based on this fact, a novel approach of deep ConvNet leaning is proposed by us.

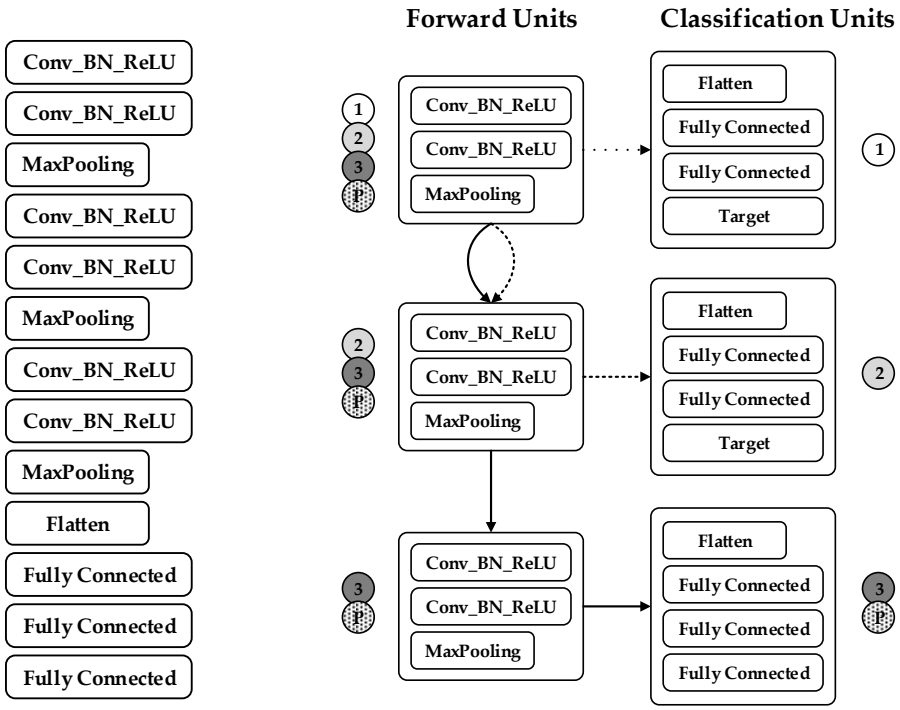
2.1 Forward Learning Convolutional Neural Network

FLCNN consists of many forward units and classification units as shown in Figure 1(b).

A forward unit usually contains convolutional layers with pooling layers and batch normalization layers to extract features from images. A classification unit has a flatten layer and fully connected layers with or without dropout, is used to perform classification.

A conventional convolutional neural network can be regard as a combination of multiple forward units and one classification unit which is shown in Figure 1(b). When the conventional CNN and our FLCNN's structure are equivalent, the testing process is totally same.

The difference between conventional ConvNet and FLCNN is the training procedure which is shown in Figure 2. Every froward unit has their corresponding classification unit. In our training process, we will train all forward units one by one. First of all, we use the first forward unit and its corresponding classification unit to build a network. Then we perform optimization. When the training procedure is done, we get a trained forward unit. After that, we frozen the weights of this trained forward unit and use it along with second forward unit and its corresponding classification unit building another network. Then we training again. Repeating this series of actions we can train all forward units. At last, we combine all trained forward units and one



(a) CNN-9 (b) FLCNN-9

Fig. 1: Conventional ConvNet and our FLCNN. (a) CNN-9, A conventional ConvNet with 6 convolutional layers and 3 fully connected layers. (b) FLCNN-9, with the similar structure of CNN-9. There are four stages for FLCNN, three training and one predicting. The circle(s) near each unit means in which stage this unit will be used. For example, the first classification unit will only be used in the first training process and all three forward units and the last classification unit will be used for prediction. The training procedure is described in Section 2.1 and Figure 2

classification unit corresponding to last forward unit to build the final network. So after the training process, all classification units except last one will be abandoned.

The structure inside forward unit or classification unit is highly customizable. In this paper, we focus more on the effectiveness of FLCNN rather than the absolute accuracy. The simplest convolutional neural network structure is enough to prove the FLCNN is effective. But of course we can use ResNeXt block [21], inception module [19, 20] and other more effective structure to construct forward units and classification units.

Because every forward unit has a corresponding classification unit and all of them except last one are not used in predicting. We could have different targets in classification unit. We also show the different targets have huge divergence of performance in section 4.2. In fact, our FLCNN is not just performing transfer learning and more than it with those changeable targets in classification units.

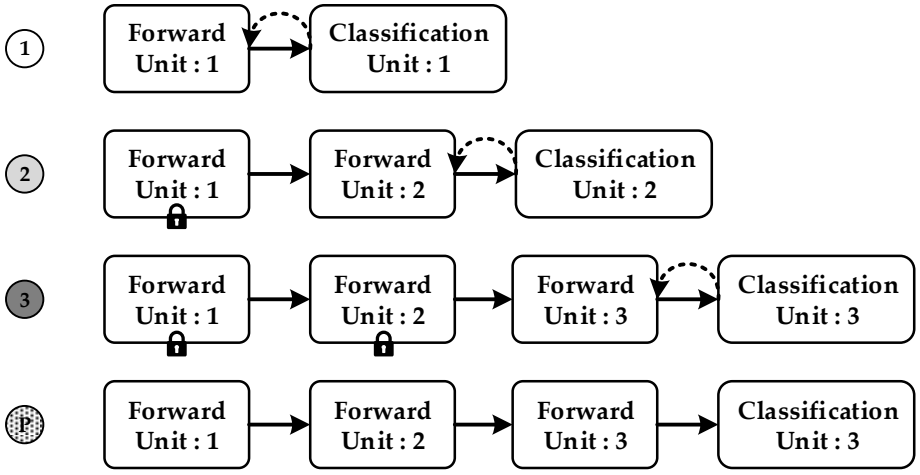


Fig. 2: FLCNN train and predict (take FLCNN-9 as example). In FLCNN-9, there are three forward units and three classification units. Training process of FLCNN-9 has three sections to train forward units one by one. In each section, the new added forward unit's weights will be updated by back-propagation. If there exists forward unit before the new added one, it will be locked. The weights of locked unit(s) will keep unchanged during training process. The predicting process will only use the last classification unit while other classification units will be abandoned.

3 Experimental Setup

3.1 Datasets

We performed our experiments with three datasets. The main dataset used in our experiments is CIFAR-10. ImageNet dataset and traffic-sign dataset are used in our different targets experiment which is describes in Section 4.2.

The CIFAR-10 dataeset is a labeled subset of the 80 million tiny images dataset which is collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. This CIFAR-10 dataset has 60000 32x32 colour images in 10 categories. Each category of dataset contains 6000 images. They have been split into 50000 training images and 10000 test images. The dataset of test part contains 1000 images random selected from each class. And the training part contains the remaining 50000 images in random order. The classes of CIFAR-10 are completely exclusive without overlap.

ImageNet is a large image dataset organized according to the WordNet hierarchy with about 150 million images in 22 thousand classes. ILSVRC is an annual competition called the ImageNet Large-Scale Visual Recognition Challenge. ILSVRC-2015 used a subset images of ImageNet which has 1000 categories and 1300 images for each category in its training set.

Traffic-sign dataset has 153 different traffic signs. There are 55726 images in train set and 27448 images in test set. The sample images is show in Figure 3(b). The average size of images is about 90x90.

(a) cifar-10 (b) traffic-signs (c) imagenet-10

Fig. 3: Samples of dataset.

3.2 Implementation Details

In this study, our programs run on a system with 2 Tesla K80 GPU. The deep learning framework we used for training is Keras with Tensorflow as backend. The version of Keras we used is 2.1.5 and the Tensorflow is 1.6.0.

Our experiments used Adam with learning rate of 0.001 which is the default value in Keras framework. We performed data augmentation in all experiments to obtain more reliable performance. For detail, we used ImageDataGenerator in Keras to achieve width shift, height shift and horizontal flip. According to the observation of our experiments, we set the training process's epoch as 100 which is enough before model reach convergence.

4 Results and Discussion

We performed three group of experiments. Section 4.1 compares the performance of convention convolutional neural networks and our forward learning convolutional neural networks. Section 4.2 presents four experiments with contrast. In each experiment, the targets of first two units are different. In section 4.3, we optimized model with ELM, and it's effective in small dataset.

4.1 Classification Units with Uniform Targets

To compare with conventional ConvNet, we train all forward units with same targets in classification units. The accuracy in validation dataset is shown in Table 1. We compared three depth of conventional ConvNet and our FLCNN. All these ConvNet have three max-pooling layers and three fully connected layers. We only count the convolutional layers and fully-connected layers as the depth of model. For example, CNN-15 has four convolutional layers before each max-pooling layer so there are 12 convolutional layers and 3 fully connected layers.

From the result of experiments, we find that the features extracted from a forward unit can be easily utilized by next forward unit. The performance in validation dataset keeps increasing with new forward units added. But with enough forward units in the network, the performance will have little improvements while the usage of computation resources keep increasing.

We also observed the degradation problem of conventional ConvNet which has been described in ResNet [8]. With the increasing depth of ConvNet, the performance will first have a improvement and then decline while our FLCNN doesn't have this problem with more stable performance

Table 1: Performance of conventional CNN and our FLCNN in CIFAR-10 dataset. The first three lines are the performance of three different layers conventional CNN. For the last three lines, we trained forward units of FLCNN one by one and in this experiment our FLCNN has three forward units, the accuracy in validation set of each training are shown in this table.

Model	1st unit	2nd unit	final
CNN-9	-	-	86.96%
CNN-15	-	-	**88.48%**
CNN-21	-	-	86.18%
FLCNN-9	77.25%	84.02%	86.55%
FLCNN-15	81.69%	85.65%	86.17%
FLCNN-21	79.89%	84.98%	**87.35%**

Table 2: We performed three experiments with the same FLCNN-21 structure and final target. The only difference is the targets of the first two classification units.

Experiment	first unit		second unit		third unit	
	target	accuracy	target	accuracy	target	accuracy
1	traffic-signs	97.29%	traffic-signs	97.38%	cifar10	66.69%
2	imagenet10	79.77%	imagenet10	89.75%	cifar10	81.05%
3	cifar10	79.89%	cifar10	84.98%	cifar10	**87.35%**

than conventional ConvNet. We analyze and think this is because our approach transfer the problem of training very deep network into several relatively shallow network. And the features from a shallow network are well utilized by another shallow network.

4.2 Classification Units with Different Targets

With multiple classification units in our FLCNN, the targets of each unit can be flexibly selected. Only the last target is decided by the problem we need to solve. In the section above, all classification units have the same targets. In this section, we show the consequence of replacing classification units' targets.

We form a image-10 dataset from 10 categories of ImageNet images in this experiment. And we also reshaped all images from different datasets into 64x64 size because our network requires a constant input dimensionality. The size of 64x64 is a trade-off between small image size (CIFAR-10 is 32x32) and large image size (ImageNet is approximately 256x256).

We use our best model in the previous section which is FLCNN-21 to experiment. The first two targets of FLCNN-21 are replaced with traffic-signs and imagenet-10.

With different targets in classification units, the results of our experiments in Table 2 show a large contrast in the final performance. Target imagenet-10 is much better than traffic-signs while both different targets are worse than uniform targets. By exploring the images of different dataset, we found imagenet-10 is more similar with cifar-10 than traffic-signs. And with the experience from transfer learning, the performance benefits from transferring features decreases when base task and target task becomes more and more dissimilar [15, 22].

4.3 Faster Solving in Small Dataset

We also try to optimize model in ELM [10] way on samll dataset. We random selected 3000 images from traffic-sign dataset and feed them into CNN and FLCNN. The architectures of two networks are show in the Table 3 and Table 4.

Table 3: CNN architecture.

type	patch size/stride	output size
input	-	91 x 91 x 3
convolution	5 x 5 / 2	44 x 44 x 96
pool	5 x 5 / 2	22 x 22 x 96
convolution	5 x 5 / 1	22 x 22 x 256
pool	5 x 5 / 2	11 x 11 x 256
convolution	3 x 3 / 1	11 x 11 x 384
fully connected	-	1 x 1 x 2048
fully connected	-	1 x 1 x 2048
fully connected	-	1 x 1 x 29

Table 4: FLCNN architecture. We simplify the network due to the shortage of memory, because least square method need to feed all data into memory during training.

type	patch size/stride	output size
input	-	91 x 91 x 3
convolution	5 x 5 / 2	44 x 44 x 96
pool	5 x 5 / 2	22 x 22 x 96
convolution	5 x 5 / 1	22 x 22 x 126
pool	5 x 5 / 2	11 x 11 x 256
convolution	3 x 3 / 1	11 x 11 x 256
fully connected	-	1 x 1 x 29

Table 5: Results of CNN and FLCNN optimized in ELM way.

model	accuracy
CNN	93.94%
FLCNN	91.20%

The results of experiment is shown in Table 5.

We perform the ELM experiment in a personal computer with only 64G memory and no GPU, so we simplify the architecture of FLCNN. In our small dataset, FLCNN optimized in ELM way is much faster than CNN-9 optimized with back-propgation while there is little worse in performance. But because it's not easy for ELM to perform batch learning and with a large dataset, the memory shortage becomes a big problem.

5 Conclusion

The results of experiments shows that our FLCNN can't replace conventional BP learning approach. But in most of cases, our FLCNN obtain similar performance compared to conventional ConvNet based on BP. So we proposed a novel learning approach to training ConvNet. With the method of training ConvNet unit by unit, we provide a way to perform assembling trained units so that transfer learning can be easily accomplished. Furthermore, FLCNN gives a platform for fast learning method like ELM which is base on least square method to be more efficient for deeper network.

Acknowledgment

This work is supported by the National Program on Key Basic Research Project (973 Program)(No. 2013CB329502).

References

1. Hamed Habibi Aghdam and Elnaz Jahani Heravi. Convolutional neural networks. 2017.
2. Les E. Atlas, Toshiteru Homma, and Robert J. Marks Ii. An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification. In *Neural Information Processing Systems, Denver, Colorado, Usa*, pages 31–40, 1987.
3. P. Le Callet, C. Viard-Gaudin, and D. Barba. A convolutional neural network approach for objective video quality assessment. *IEEE Transactions on Neural Networks*, 17(5):1316–27, 2006.
4. D. S. Clouse, C. L. Giles, B. G. Horne, and G. W. Cottrell. Time-delay neural networks: representation and induction of finite-state machines. *IEEE Transactions on Neural Networks*, 8(5):1065–70, 1997.
5. Ciregan Dan, Ueli Meier, and Jrgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.
6. Patrick O. Glauner. Deep convolutional neural networks for smile recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 22(10):1533–1545, 2015.
7. S Haykin and B Kosko. *GradientBased Learning Applied to Document Recognition*. PhD thesis, Wiley-IEEE Press, 2009.
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
9. Hong Hu, Liang Pang, Dongping Tian, and Zhongzhi Shi. Perception granular computing in visual haze-free task. *Expert Systems with Applications*, 41(6):2729–2741, 2014.
10. Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.
11. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the Acm*, 60(2):2012, 2012.
12. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
13. Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2001.
14. Vernon B Mountcastle. The columnar organization of the neocortex. *Brain: a journal of neurology*, 120(4):701–722, 1997.
15. Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
16. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning representations by back-propagating errors*. MIT Press, 1988.

17. David E. Rumelhart, James L. Mcclelland, and The Pdp Group. Parallel distributed processing: Foundations v. 1: Explorations in the microstructure of cognition. *Language*, 63(4):45 – 76, 1986.
18. J. Schmidhuber, U. Meier, and D. Ciresan. Multi-column deep neural networks for image classification. 157(10):3642–3649, 2012.
19. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.
20. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.
21. Saining Xie, Ross Girshick, Piotr Dollr, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.
22. Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.