INTERNATIONAL JOURNAL OF TECHNO-ENGINEERING
IJTE

ISSN: 2057-5688

Object Recognition and Detection Using Convolutional Neural Network in Deep Learning

E. Akhil Babu [1], B. Govardhan [2]
[1] Assistant professor, Associate Professor [2]
Department of Computer Science Engineering
RISE Krishna Sai Prakasam Group of Institutions

Abstract— Object recognition technology in the field of computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, although the image of the objects may vary somewhat in different viewpoints, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades In many computer vision systems, object detection is the first task being performed as it allows us to obtain further information regarding the detected object and about the scene. In recent years, the use of deep learning has attracted the attention of researchers, Deep learning uses multiple layers to extract raw features from high-level features the machine divides each complex concept into simpler concepts. The proposed method is based on AlexNet architecture. In this method, a convolutional neural network-based architecture with a small amount of data can detect objects in the image, the data used is the March database, so this process identifies thirty pictures in four separate classes with 100% accuracy. Keywords— Computer Vision, Deep Learning, Convolutional Neural Network, Object.

1. Introduction

There are fascinating problems with computer vision, such as image classification and object detection, both of which are part of an area called object recognition. For these types of issues, there has been a robust scientific development in the last years, mainly due to the advances of convolutional neural networks, deep learning techniques, and the increase of the parallelism processing power offered by the graphics processing units (GPUs). The image classification problem is the task of assigning to an input image one label from a fixed set of categories. This classification problem is central within computer vision because, despite its simplicity, there are a wide variety of practical applications and has multiple uses, such as labeling skin cancer images [1], use of high-resolution images to detect natural disasters such as floods, volcanoes, and severe droughts, noting the impacts and damage caused [2–4]. The performance of image classification

ISSN: 2057-5688

algorithms crucially relies on the features used to feed them [5]. It means that the progress of image classification techniques using machine learning relied heavily on the engineering of selecting the essential features of the images that make up the database. Thus, obtaining these resources has become a daunting task, resulting in increased complexity and computational cost. Commonly, two independent steps are required for image classification, feature extraction, and learning algorithm choice, and this has been widely developed and enhanced using support vector machines (SVMs). The SVM algorithm, when considered as part of the supervised learning approach, is often used for tasks as classification, regression, and outlier detection [6]. The most attractive feature of this algorithm is that its learning mechanism for multiple objects is simpler to be analyzed mathematically than traditional neural network architecture, thus allowing to complex alterations with known effects on the core features of the algorithm [7]. In essence, an SVM maps the training data to higher-dimensional feature space and constructs a separation hyperplane with maximum margin, producing a nonlinear separation boundary in the input space [8]. Today, the most robust object classification and detection algorithms use deep learning architectures, with many specialized layers for automating the filtering and feature extraction process. Machine learning algorithms such as linear regression, support vector machines, and decision trees all have its peculiarities in the learning process, but fundamentally they all apply similar steps: make a prediction, receive a correction, and adjust the prediction mechanism based on the correction, at a high level, making it quite similar to how a human learns. Deep learning has appeared bringing a new approach to the problem, which attempted to overcome previous drawbacks by learning abstraction in data following a stratified description paradigm based on a nonlinear transformation [9]. A key advantage of deep learning is its ability to perform semi-supervised or unsupervised feature extraction over massive datasets. The ability to learn the feature extraction step present in deep learning-based algorithms comes from the extensive use of convolutional neural networks (ConvNet or CNN). In this context, convolution is a specialized type of linear operation and can be seen as the simple application of a filter to a determined input [10]. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in the input by tweaking the parameters of the convolution. The network can adjust itself to reduce the error and therefore learn the best parameters to extract relevant information on the database. Many deep neural network (DNN)-based object detectors have been proposed in the last few years [11, 12]. This research investigates the performance of stateof-the-art DNN models of SSD and Faster RCNN applied to a classical detection problem where the algorithms were trained

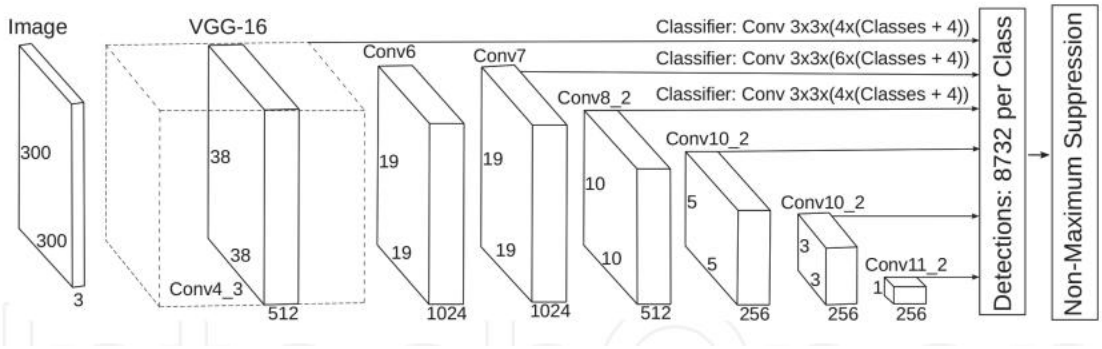
ISSN: 2057-5688

to identify several animals in images; furthermore to exemplify the application in scientific research, the YOLO network was trained to solve the mice tracking problem. The flowing sections describe the DNN models mentioned earlier in more details [13–15].

2. Object detection techniques

2.1 Single shot multibox detector

The single shot multibox detector [13] is one of the best detectors in terms of speed and accuracy comprising two main steps, feature map extraction and convolutional filter applications, to detect objects. The SSD architecture builds on the VGG-16 network [16], and this choice was made based on the strong performance in high-quality image classification tasks and the popularity of the network in problems where transfer learning is involved.



Instead of the original VGG fully connected layers, a set of auxiliary convolutional layers change the model, thus enabling to extract features at multiple scales and progressively decrease the size of the input to each subsequent layer. The bounding box generation considers the application of matching precomputed, fixed-size bounding boxes called priors with the original distribution of ground truth boxes. These priors are selected to keep the intersection over union (IoU) ratio equal to or greater than 0:5. The overall loss function defined in Eq. (1) is a linear combination of the confidence loss, which measures how confident the network is of the computed bounding box using categorical cross-entropy and location loss, which measures how far away the networks predicted bounding boxes are from the ground truth ones using L2 norm.

$$L(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)\right)$$

where N is the number of matched default boxes and Lconf and Lloc are the confidence and location loss, respectively, as defined in [13]. Figure 1 depicts how to apply the convolutional kernels to an input image in the SSD architecture.

2.2 You only look once

ISSN: 2057-5688

You only look once [14] is a state-of-the-art object detection algorithm which targets real-time applications, and unlike some of the competitors, it is not a traditional classifier purposed as an object detector. YOLO works by dividing the input image into a grid of S S cells, where each of these cells is responsible for five bounding boxes predictions that describe the rectangle around the object. It also outputs a confidence score, which is a measure of the certainty that an object was enclosed. Therefore the score does not have any relation with the kind of object present in the box, only with the box's shape. For each predicted bounding box, a class it's also predicted working just like a regular classifier giving resulting in a probability distribution over all the possible classes. The confidence score for the bounding box and the class prediction combines into one final score that specifies the probability for each box includes a specific type of object. Given these design choices, most of the boxes will have low confidence scores, so only the boxes whose final score is beyond a threshold are kept. Eq. (2) states the loss function minimized by the training step in the YOLO algorithm.

$$\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]$$
$$+\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$
$$+\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\sum_{i=0}^{s^2}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2$$

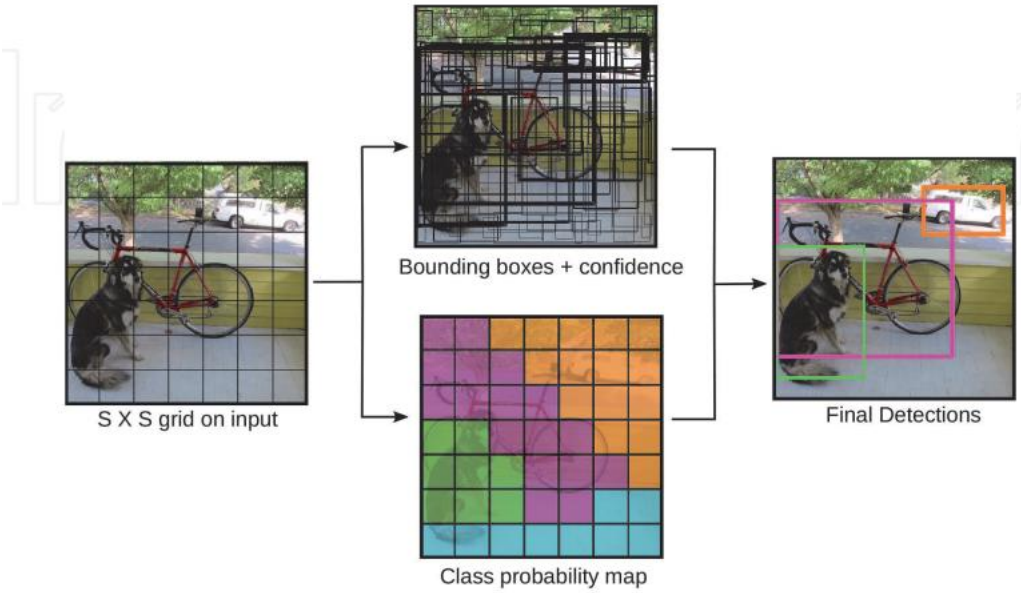
where 1obj i indicates if an object appears in cell i and 1obj ij denotes the j th bounding box predictor in cell i responsible for that prediction; x, y, w, h, and C denote the coordinates that represent the center of the box relative to the bounds of the grid cell. The width and height predictions are relative to the whole image. Finally, C denotes the confidence prediction, that is, the IoU between the predicted box and any ground truth box. Figure 2 describes how the YOLO network process as image. Initially, the input gets passed through a CNN producing the bounding boxes with its perspectives confidences scores and generating the class probability map. Finally, the results of the previous steps are combined to form the final predictions.

2.3 Faster region convolutional neural network

The faster region convolutional neural network [15] is another state-of-the-art CNN-based deep learning object detection approach. In this architecture, the network takes the provided input image into a convolutional network which provides a convolutional feature map. Instead of using the selective search algorithm to identify the region proposals made in previous iterations [18,

ISSN: 2057-5688

19], a separate network is used to learn and predict these regions. The predicted region proposals are then reshaped using a region of interest (ROI) pooling layer, which is then used to



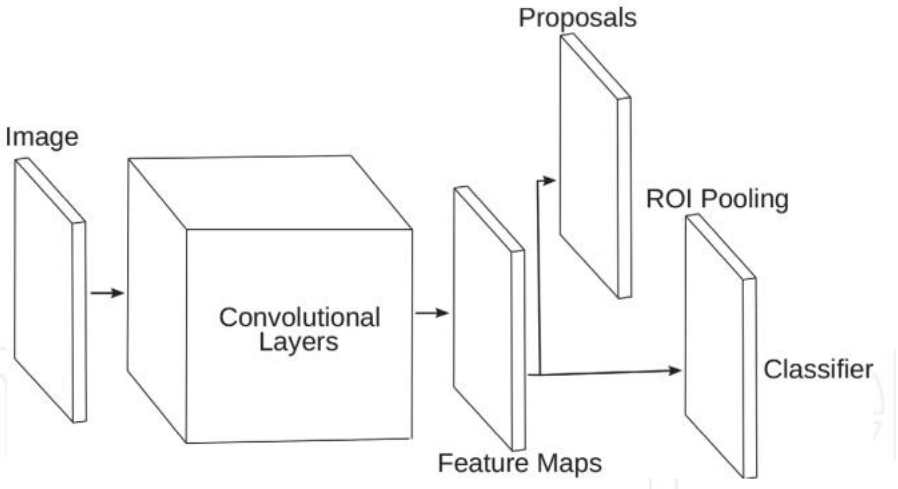


Figure 3. Faster RCNN acts as a single, unified network for object detection [15]. The region proposal network module serves as the "attention" of this unified network. Figure based on [15].

classify the image within the proposed region and predict the offset values for the bounding boxes. The strategy behind the region proposal network (RPN) training is to use a binary label for each anchor, so the number one will represent the presence of an object and number zero the absence; with this strategy any IoU over 0:7 determines the object's presence and below 0:3

ISSN: 2057-5688

indicates the object's absence. Thus a multitask loss function shown in Eq. (3) is minimized during the training phase.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*)$$

where i is the index of the anchor in the batch, pi is the predicted probability of being an object, p∗ i is the ground truth probability of the anchor, ti is the predicted bounding box coordinate, t ∗ i is the ground truth bounding box coordinate, and Lcls and Lreg are the classification and regression loss, respectively. Figure 3 depicts the unified network for object detection implemented in the Faster RCNN architecture. Using the recently popular terminology of neural networks with "attention" mechanisms [20], the region proposal network module tells the Fast RCNN module where to look [15].

3. Datasets

A sample of the PASCAL VOC [21] dataset is used to exemplify the use of SSD and RCNN object detection algorithms. A sample of 6 classes of the 20 available were selected. Table 1 describes the sample size selected for each class. The images presented in the dataset were randomly divided as follows: 1911 for training corresponding to 50%, 1126 for validation corresponding to 25% and test also corresponding to 25%. To further illustrate the applications of such algorithms in scientific research, the dataset used for the YOLO network presented in [22] was also analyzed. As described in [22], the dataset is composed of images from three researches that involve behavioral experiments with mice:

• Ethological evaluation [23]: This research presents new metrics for chronic stress models of social defeat in mice.

• Automated home-cage [24]: This study introduces a trainable computer vision system that allows the automated analysis of complex mouse behaviors; they are eat, drink, groom, hang, micromovement, rear, rest, and walk.

• Caltech Resident-Intruder Mouse dataset (CRIM13) [25]: It has videos recorded with superior and synchronized lateral visualization of pairs of mice involved in social behavior in 13 different actions. Table 2 describes the sample size selected from each of the datasets used in this paper. For the ethological evaluation [23], 3707 frames were used, captured in a top view of the arena of social interaction experiments among mice. For the automated home-cage [24], a sample of 3073 frames was selected from a side view of behavioral experiments. For the CRIM13 [25], a sample of 6842 frames was selected, 3492 from a side view and 3350 from a top view. The same dataset

ISSN: 2057-5688

division used in [22] was also reproduced resulting in 6811 images for training, 3405 for validation, and 3406 for the test.

4. Material and methods for object detection

In this work, the previously described SDD and Faster RCNN networks are compared in the task of localization and tracking of six species of animals in diversified environments. Having accurate, detailed, and up-to-date information about the location and behavior of animals in the wild would improve our ability to study and conserve ecosystems [26]. Additionally, results from the YOLO network, reproduced from [22], to detect and track mice in videos are recorded during behavioral neuroscience experiments. The task of mice detection consists of determining the location in the image where the animals are present, for each frame acquired. The computational development here presented was performed on a computer with CPU AMD Athlon II X2 B22 at 2:8GHz, 8GB of RAM, NVIDIA GeForce GTX 1070 8GB GPU, Ubuntu 18:04 LTS as OS, CUDA 9, and CuDNN 7. Our approach used the convolutional networks described in Section 2.

5. Results and conclusion

The results obtained for the SSD and Faster RCNN networks in the experiments were based on the analysis of 4163 images, organized according to the dataset described in Section 3. Figure 4(a) depicts the increasing development of the mean average precision values in the epochs of training. Both architectures reached high mean average precision (mAP) while successfully minimizing the values of their respective loss functions. The Faster RCNN network presented higher and better stability in precision, which can be seen by the smoothness in its curve. Figure 4(b) is a box plot of the time spent by each network on the classification of a single image, whereas the SSD came ahead with 17 2 ms as the mean and standard deviation values, and the Faster RCNN translated its higher computational complexity in the execution time with 30 2ms as the mean and standard deviation values, respectively. Table 3 presents more results related to object detection performance. First, it shows the mean average precision, which is the mean value of the average precisions for each class, where average precision is the average value of 11 points on the precision-recall curve for each possible threshold, that is, all the probability of detection for the same class (Precision-Recall evaluation according to the terms described in the PASCAL VOC [21]). Figure 5 shows some selected examples of object detection results on the dataset used. Each output box is associated with a category label and a softmax score in ½0, 1→ A score threshold of 0:5 is used to display these images.

ISSN: 2057-5688

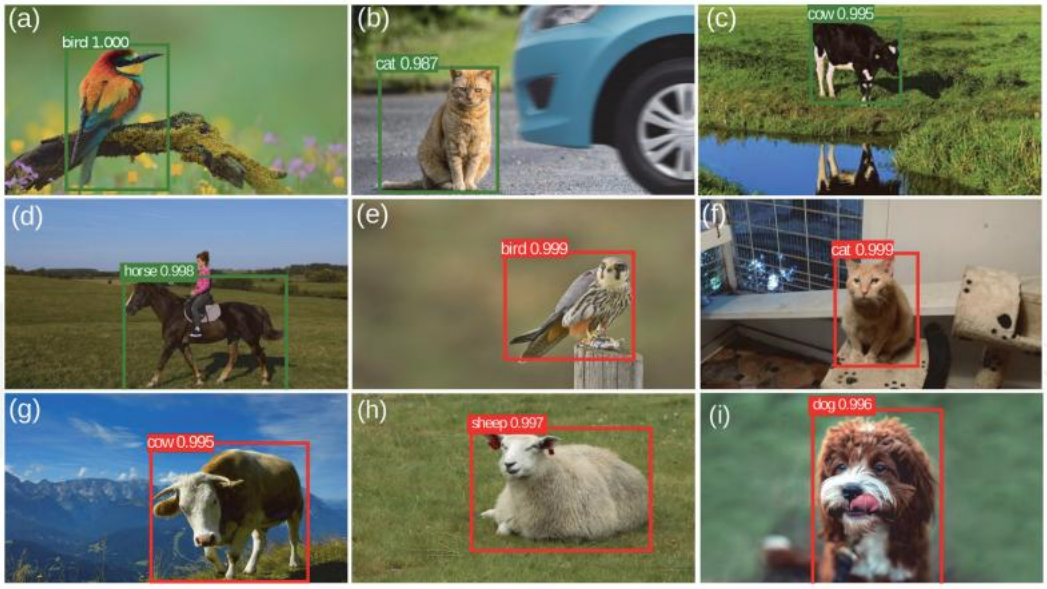


Figure 5. Output examples of the networks. (a)–(d) refer to SSD and (e)–(i) to Faster RCNN.

Our approach, as in [22], also used two versions of the YOLO network to detect mice within three different experimental setups. The results obtained were based on the analysis of 13,622 images, organized according to the dataset described in Section 3. The first version of YOLO being trained was the YOLO Full network which uses the Darknet-53 [14] convolutional architecture that comprises 53 convolutional layers. Such a model was trained as described in [17], starting from an ImageNet [28] pre-trained model. Each model requires 235 MB of storage size. We used a batch of eight images, a momentum of 0:9, and weight decay. The model took 140 hours to be trained. A smaller and faster YOLO alternative was also trained and named as YOLO Tiny. To speed up the process, this "tiny" version comprises only a portion of the Darknet-53 [14] resources: 23 convolutional layers. Each model requires only 34 MB of storage size. The network training follows as described in [17], finetuning an ImageNet [28] pre-trained model. We used a batch of 64 images, a momentum of 0:9, and weight decay. The model took 18 hours to be trained. Figure 6 shows the comparison of the two YOLO models used, YOLO Full and Tiny. Figure 6(a) shows high accuracy of the Full architecture with small oscillations of the accuracy curve during the training. In Figure 6(b), the high accuracy is maintained from the earliest times and remains practically unchanged up to the limit number of epochs. Both architectures reached high mean average precision values while successfully minimizing the values of their loss function. The Tiny version of the YOLO network presented better stability in precision, which can be seen by the smoothness in its curve.

ISSN: 2057-5688

The results show that the mean average precision reached by this re-implementation was 90.79 and 90.75% for the Full and Tiny versions of YOLO, respectively. The use of the Tiny version is a good alternative for experimental designs that require real-time response. Figure 6(c) is a bar graph showing the mean time spent on the classification of a single image in both architectures. The smaller size of the Tiny version gets a direct translation in execution time, having 0:08 0:06s as the mean and standard deviation values, whereas the Full version has 0:36 0:16s as the mean and standard deviation values, respectively.

References

[1] Esteva A et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 2017; 542(7639):115

[2] Jayaraman V, Chandrasekhar MG, Rao UR. Managing the natural disasters from space technology inputs. Acta Astronautica. 1997;40(2–8):291-325

[3] Leonard M et al. A compound event framework for understanding extreme impacts. Wiley Interdisciplinary Reviews: Climate Change. 2014;5(1): 113-128

[4] Kogan FN. Global drought watch from space. Bulletin of the American Meteorological Society. 1997;78(4): 621-636

[5] Srinivas S, Sarvadevabhatla RK, Mopuri RK, Prabhu N, Kruthiventi SSS, Venkatesh Babu R. An introduction to deep convolutional neural nets for computer vision. In: Deep Learning for Medical Image Analysis. Academic Press; 2017. pp. 25-52

[6] de Menezes RST, de Azevedo Lima L, Santana O, Henriques-Alves AM, Santa Cruz RM, Maia H. Classification of mice head orientation using support vector machine and histogram of oriented gradients features. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE; 2018. pp. 1-6

[7] Oskoei MA, Gan JQ, Hu H. Adaptive schemes applied to online SVM for BCI data classification. In: 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE; 2009. pp. 2600-2603

[8] Hearst MA, Dumais ST, Osuna E, Platt J, Scholkopf B. Support vector machines. IEEE Intelligent Systems and their Applications. 1998;13(4):1828

[9] Pan WD, Dong Y, Wu D. Classification of malaria-infected cells using deep convolutional neural networks. In: Machine Learning: Advanced Techniques and Emerging Applications. 2018. p. 159

[10] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016

[11] Deng L, Hinton G, Kingsbury B. New types of deep neural network learning for

ISSN: 2057-5688

speech recognition and related applications: An overview. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE; 2013. pp. 8599-8603

[12] Kriegeskorte N. Deep neural networks: A new framework for modeling biological vision and brain information processing. Annual Review of Vision Science. 2015;1:417-446

[13] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, et al. SSD: Single shot multibox detector. In: European Conference on Computer Vision. Cham: Springer; 2016. pp. 21-37

[14] Redmon J, Farhadi A. Yolov3: An Incremental Improvement. arXiv; 2018

[15] Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. 2015. pp. 91-99

[16] Simonyan K, Zisserman A. Very deep convolutional networks for largescale image recognition. arXiv preprint arXiv:1409.1556; 2014.