A Deep Learning Approach for Vehicle Detection

Mohamed Ashraf Ali[1], Hossam E. Abd El Munim[2], Ahmed Hassan Yousef[2,3], and Sherif Hammad[1]
[1]Department of Mechatronics Engineering, Ain Shams University, Cairo, Egypt
[2]Department of Computer and Systems Engineering, Ain Shams University, Cairo, Egypt
[3]School of Information Technology and Computer Science, Nile University, Giza, Egypt

***Abstract*—The autonomous driving needs some several features to achieve driving without human interference. One of these features is vehicle classification and detection since the target of this process is to help the CPU "Central Processing Unit" of the vehicle to see what is around the vehicle, in order to evaluate the situation to take the best decision for each situation in real time. This paper is focusing on the classification process of the video-based vehicle detection. to achieve that, different deep learning techniques have been implemented which are known as convolutional neural networks (CNN) architectures. These CNN architectures are ResNet, Inception-ResnetV2,InceptionV3, NASNet, MobileNetV2, and PNASNet architectures. Also there are two different datasets have been trained in these architectures to evaluate them. These datasets are Kitti dataset to train on car detection only, in additions to MIO-TCD dataset to detect various types of vehicles. The Inception-ResnetV2 have shown the best performance in our results.**

***Keywords*—Deep-learning; CNN; ResNet; InceptionV3; Inception-ResnetV; MobileNetV2; NASNet; PNASNet; Vehicle-detection; classification; Kitti; MIO-TCD dataset.**

I. Introduction

Autonomous driving is not a dream anymore. The industry of vehicles already entered the race of building systems that can see around, sense its relative speed, evaluate the situation, and take the suitable decision in real time. There was a lot of ambitious trials from the 1920s and experiments at 1950s to achieve this target, but the main obstacle was the power and the performance of the control systems. After that in the 1990s when the efficiency of CPUs start growing, the dream of autonomous vehicles started to glow again in researchers minds by trying to combine sensors, cameras, and radars with the driving systems but they cannot get the full power of the computer vision systems yet. On the other hand, machine learning was first introduced by Alan Turing in the 1940s [1]. Alan Turing's paper focused on the machines that could be intelligent by checking how the brain works and trying to simulate the same idea on machines. Many scientists and researchers tried to make progress in the machine learning algorithms. But the first CNN architecture was introduced by Yann LeCun, and he called it LENET5 [2]. Of course in these days there was no Graphical Processing Unit "GPU" available, and even all of the CPUs was slow, but this network was the key and the start of the CNN architectures. Since then to the year 2010, there was a gap in the development of CNN architectures because there was not enough computing power to help in faster computing.

Meanwhile, the Vehicle detection was based on classical methods like: HOG with SVM and sliding window search [3], Ada-boost [4] till 2010. Dan Claudiu Ciresan and Jurgen Schmidhuber implemented a first neural network with GPUs [5]. They used NVIDIA GTX 280 with only 9 layers in this network structure. But the power of the deep learning in image classification and detection did not show up until 2012. The first CNN architecture won the 2012 ImageNet LSVRC-2012 competition with 15.3% error rate. The second place had 26.2% error rate. This CNN architecture was called AlexNet [6] with only 5 convolutional layers and 3 fully connected layers. AlexNet dragged the researchers' attention to produce more efficient architectures. Since then, CNN architectures won the competition every year, with lower error rates than before. Given that the human error rate is 5%, ResNet [7] surprisingly won the ImageNet competition with an error rate of 3.6%.

From 2012 until now, deep learning and CNN architectures entered the industry field [8]. Deep learning could be used in chain lines, robotics, healthcare, and retails. Also deep learning had entered the autonomous vehicle field by making object detection, depth estimation, and face recognition. The automation companies now are working hard in deep learning field in order to achieve the best results with the minimum error rate, to make the best safety performance. Likewise, machine learning and deep learning have been used by a lot of big companies like Facebook for promotions analysis and face recognition, Google for search and visiting analysis, and in google maps, Tesla for making Autonomous vehicle, Siemens for improving the efficiency for products and construction, Fanuc for making their robots smarter, and finally KUKA for making robots too. And now, new companies and startups in deep learning and AI field are being established each day because this field can propose new solutions to the obstacles that facing humanity progress.

In this paper, we will discuss the difference in accuracy of the classification inside the detection process between six CNN architectures, and implement two of them with different two detection systems to know which one gives the best performance. This paper is divided into two main sections, the first section discuss the methods in this paper. It shows the difference between 6 CNN architectures. Then the second part shows the results on two different datasets to reveal which one has the best performance among others.

II. Methods

A. General object detection algorithm

To understand how the object detection algorithm works, we need first to know the difference between classification and detection. The classification is classified the input image into its corresponding category, but the detection finds and localizes the object of interest inside the image.

The generalized object detection algorithm consists of three main parts as shown in Fig.1

1) In the first step, the algorithm will choose which regions in the input image will enter the next step. It's called region proposal step. There are several ways to make this task. Like in faster R-CNN [9] the region proposals and the feature extractors are combined by using "region proposal networks" to choose regions. But in SSD [10], it uses "The bounding box regression", by making a combination of default different sizes boxes to the input feature map. Finally in YOLO [11] it separates the input image into a fixed number of grids.
2) Then, the algorithm uses a certain feature extractor to extract any needed features, and get the classification score for each of the proposed regions "or grids" using a certain CNN architecture.
3) Finally, trying to combine any overlapping boxes in the result into a single box for each object using a certain algorithm like "non-maximum suppression".

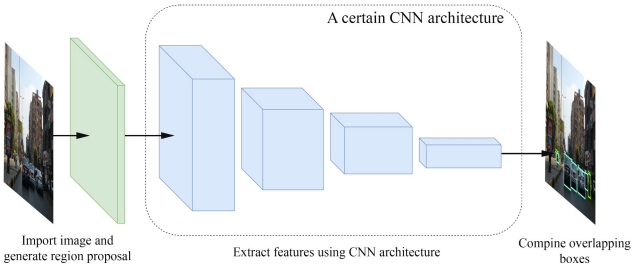

Fig. 1. The components of the generalized object detection algorithm.

In the next section, we will focus on the second step of the vehicle detection, by trying to implement a certain number of CNN architectures, to make the comparison between them.

B. CNN architectures

1) ***ResNet****:* The Residual Networks "Resnet" was invented by Microsoft researchers and first introduced in ILSVRC'15 which won the first place with 3.57% error in the classification process. It was established to make deeper neural networks (from tens of layers to hundreds of layers) to learn more advanced and more complex functions and features in the classification process. Resnet is consists of residual blocks which made to overcome the high error rates while training deep networks, the idea of resnet blocks is getting the input before the block and add it with the result of Convs layers before the ReLU nonlinear function to adjust the featured result to make better performance by making shortcuts in the network. The Resnet Block made from 2 layers in the small versions and 3 layers in the bigger ones.

2) ***Inception****:* Inception network [12] was first introduced in 2014 by Google's researchers, which won the ImageNet competition in the classification process. This model is based on a principle called "Inception cell" by making and repeating series of convolutional layers of different sizes and different filter sizes. In Inception V1 the filter sizes were from 1x1 to

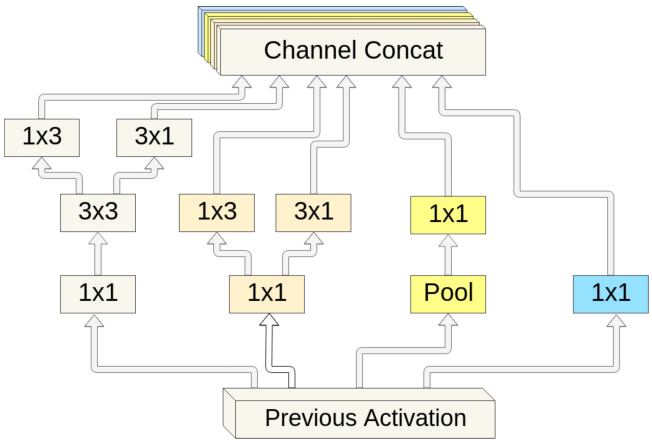

Fig. 2. The components of the Inception block. Inception-Resnet V2

5x5 filters to extract features from each filter to get features as much as it can. But the researchers found that they can make two 3X3 filters instead of 5x5 to minimize the number of parameters and reduce the computation time, then all of these features assembled together in one layer. And finally in InceptionV2 and InceptionV3, Google researchers have replaced one of the 3x3 filters with 3x1 and 1x3 filters as shown in Fig.2 for the best result at this time. To improve the performance of the network, the researchers made 2 auxiliary outputs in the network. Just to regularize the output of the network.

3) ***Inception-Resnet****:* This network was the result of the cooperation between the authors of Inception architectures, and the authors of ResNet architectures, by taking the best of both networks and combine them together in one network. This network as shown in fig.3 is consists of a lot of inception layers and combined with the Residual connection from the previous layer. The authors said that the new network can make the training process faster. They made some versions of this idea like Inception V4, Inception-Resnet V1 and Inception-Resnet V2. Inception-Resnet V2 [13] has been chosen to implement in this paper because the authors said that the Inception-Resnet V2 is significantly better than Inception-Resnet V1 in the recognition tasks.

4) ***MobileNetV2****:* In April 2017, Google researchers had introduced a brand new architecture called MobileNet [14]. They said that MobileNet was invented to make the classification task using small devices like phones faster than any other architectures at this moment, also it's had the same accurate like larger convolutional networks. The mobilenetV1 have 3x3 depthwise convolutional layer with a 1x1 Pointwise

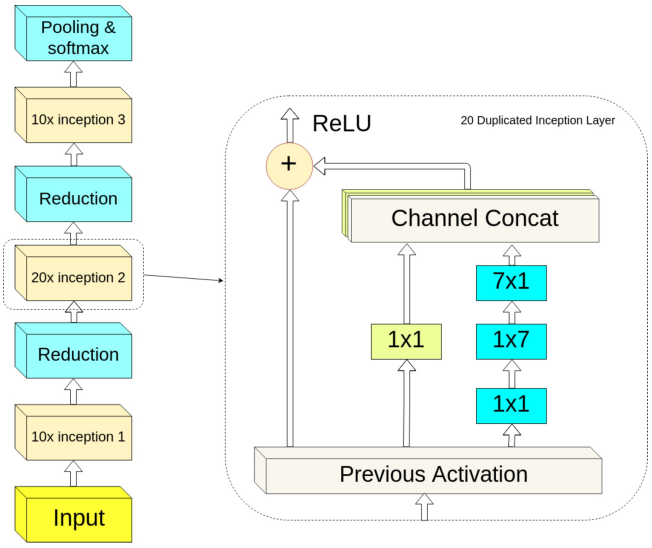

Fig. 3. The components of the Inception-Resnet architecture.

convolutional layer with ReLU activation function duplicated by 13 times. These layers make the same job as the traditional convolutional layer, but it is faster.

MobileNetV2 [15] uses the same ideas as MobileNetV1 with two new things: using linear bottleblocks, and using Residual connection in each layer like shown in Fig.4, they also replaced the big convolutional layers with smaller ones, even if the result was adding more layers, the shortcuts between layers can make this network more efficient with less number of parameters compared with MobileNetV1. The 1x1 Expansion layer was added with 1x1 projection layer instead of the 1x1 pointwise layer which is used in MobileNetV1, also a residual connection was added.

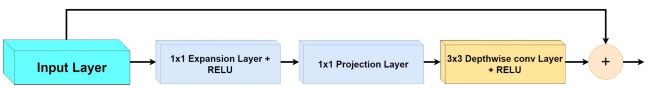

Fig. 4. The components of the MobileNetV2 layer block.

5) ***NASNet****:* In 2017, Google researchers had introduced a new way to build ConvNets, it is called "AutoML". This approach is based on making a small architecture for a specific task, and trying to automatically modify the architecture's block based on the feedback which comes from the output of the architecture "error rate" by RNN method to predict which nodes in the block should be modified, and repeat this process as long as they can to reach the best block components and reach best accuracy with the lowest number of the parameters. Then after the AutoML reaches the best block components, the researchers took the block and duplicate it to add more parameters to make bigger tasks than the small one. In the classification task, the CIFAR10 has been chosen for building the architecture's block and then implement the same block components to make NASNet-large [16] for bigger datasets like COCO. If they tried to make "AutoML" process with COCO dataset directly, the training process will take several months because the COCO dataset has more than 1000 class.

6) ***PNASNet****:* The CNN architecture PNASNet [17] has the same idea of NASNet except it uses "Sequential Model-Based Optimization (SMBO)" instead of RNN from NASNet. The PNASNet authors claim that the new method will produce blocks more efficient and faster than NASNet. They are trying to convert this architecture from a simple way to a progressive way by making it more complex.

III. Experiments and Results

A. Hardware and Software Requirements

The training process of deep learning architectures and convolutional neural networks for image classifications and detection tasks needs a lot of computational power to get acceptable results. The training on CPU only will take weeks to finish the training for one model only. That is why any deep learning research must need at least one powerful GPU or more to finish the training process in a short time compared with CPU training (hours). The GPU NVIDIA GeForce GTX 1060 6 GB version has been chosen for this task because it has an acceptable power to finish the training process with an acceptable price. It comes with 1280 cuda cores, memory bandwidth 192 Gb/sec, and it has 6 GB of RAM to take more than 50 images per one batch without memory fail, also this GPU has been supported with CPU Intel i5-8400 and 16 GB of Ram to control, execute, and move images from hard drive to the memory of the GPU.

We used the Linux based operating system (Ubuntu 18.04) with python 3 in the training and evaluation process. Python programming language had been used because it has a big support to the matrix and deep learning libraries. Also, the Nvidia parallel programming language CUDA had been used to control our GPU to implement the training and evaluation process using GPU-accelerated processes. Finally Tensorflow-GPU 1.9 had been used while implementing the deep learning models. Also, TensorBoard is a toolkit used for making charts while making the training process to track the progress of the training. Real-time charts have been made with TensorBoard between the training and validation error with the number of the training steps. TensorBoard is used in the training process to avoid overfitting while training the model.

B. Training on Kitti dataset

Kitti dataset was made by "Karlsruhe Institute of Technology" in association with Toyota, it contains more than 7,500 colored images for the training, which captured from roads and highways from the United States. They used their own cars to capture these images to be an on-road dataset and to be valid to be used on autonomous vehicles researches. This dataset was made to evaluate and compare between vehicle detection methods to rank these methods. The first image in Fig.5 has no vehicle to classify, so the true positive in this photo should be "No Vehicle" class. The second and third image contains vehicles on them, so the true positive in these images should be "Vehicle" class. We evaluated these images

TABLE I
THE COMPARISON BETWEEN CNN ARCHITECTURES ON KITTI DATASET USING ONLY TWO CLASSES (VEHICLE, NO VEHICLE).

Architecture	True Positive rate
InceptionV3	91.8%
RESNET	89.1%
Inception-ResnetV2	91.4%
MobileNetV2	90.8%
NASNET	92.3%
PNASNET	90.5%

using the discussed CNN architectures in this paper with the true positive value for each architecture.

ResNet =90% MobileNetV2 =90%
InceptionV3 =95% NASNet =94%
Inception-ResnetV2 =92% PNASNet =91%

ResNet =95% MobileNetV2 =94%
InceptionV3 =94% NASNet =97%
Inception-ResnetV2 =96% PNASNet =94%

ResNet =93% MobileNetV2 =93%
InceptionV3 =94% NASNet =96%
Inception-ResnetV2 =94% PNASNet =95%

Fig. 5. The true positive percentage results of the cited CNN architectures in this paper.

We used Kitti dataset to compare CNN architectures in the detection task, especially on the classification process. We compared these CNN architectures in just two classes: "vehicle, and No vehicle". As shown in Table.I, the difference between each CNN architecture is not high, the difference between the best one (NASNet architecture) and the worse one (ResNet architecture) did not exceed 3.2%. NASNET architecture and InceptionV3 have shown great and accurate results.

C. Training on MIO-TCD dataset

MIO-TCD dataset is a brand new dataset for different classes on road. They used thousands of cameras in the United States and Canada to collect images from various conditions and time to cover most of the traffic scenarios. This dataset contains more than 600,000 different images for 11 classes from the roads: articulated truck, pedestrian, bicycle, pickup truck, bus, single unit truck, car, work van, motorcycle, Non-motorized vehicle, and background. MIO-TCD dataset had been divided into two groups. Classification challenge with more than 645,000 images and localization challenge with more than 120,000 images. This dataset had been chosen because the classification images had been sliced from images which is the same process that some of detections systems do like YOLO [14]. So because we want to evaluate the classification process only in the vehicle detection task, this dataset is one of the best choices to achieve this target. The table.2 shown the comparison between the 6 CNN architectures in the true positive results (top-1 accuracy) of the testing images from MIO-TCD dataset. As shown in table II, Inception-ResnetV2 has shown a good top-1 accuracy among other CNN architectures, Inception-ResnetV2 suppressed the other architectures with margin 1.3%. On the other hand, PNASNet was the worse one between them. Also, some classes have an accuracy over 80% in all of the architectures like cars and bus, and in "background" class most architectures did not exceed 70%, since the cars and buses have some common features. But the background is hard to the architecture to get common features to fine-tuning weights to detect them.

D. Vehicle detection algorithms results

The next two images contain the result of the detection process using two different vehicle detection algorithms, the first one shows the object detection result from Faster R-CNN with Inception architecture as a feature extractor, and the second one shows the object detection result from SSD with MobileNet as a feature extractor also. As shown in Fig

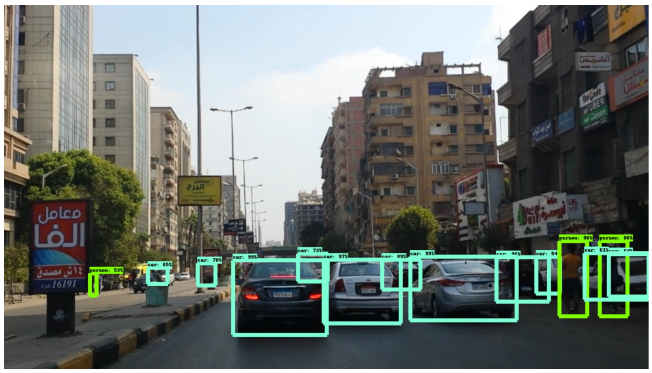

Fig. 6. The detection result from Faster R-CNN with Inception architecture.

6 and Fig 7 the results using Faster R-CNN is better than SSD algorithm, however, the difference in the detection speed is also big. SSD can make the detection by 5-6 frames per second using Nvidia GTX 750, but Faster R-CNN makes only

TABLE II
THE COMPARISON BETWEEN CNN ARCHITECTURES ON MIO-TCD DATASET.

	Inception V3	RESNET	Inception-ResnetV2	MobileNetV2	NASNet	PNASNet
Articulated truck	75.2%	77.5%	79.0%	78.7%	74.6%	74.2%
Bicycle	75.5%	74.8%	78.0%	79.5%	74.4%	75.3%
Bus	83.3%	85.5%	86.5%	84.3%	84.5%	82.5%
Car	79.0%	81.0%	86.6%	83.7%	80.5%	81.5%
Motorcycle	74.4%	77.3%	73.5%	75.3%	74.5%	74.0%
Non-motorized vehicle	72.5%	71.8%	74.3%	73.5%	72.3%	70.5%
Pedestrian	82.0%	83.5%	84.4%	83.0%	80.2%	78.9%
Pickup truck	74.0%	72.9%	76.3%	73.8%	72.8%	73.5%
Single unit truck	72.5%	74.9%	77.7%	75.3%	76.3%	72.5%
Work van	71.3%	77.0%	78.8%	74.5%	73.6%	72.2%
Background	67.2%	68.4%	71.3%	70.3%	69.5%	66.6%
Average score	**75.17%**	**76.78%**	**78.76%**	**77.45%**	**75.75%**	**74.70%**

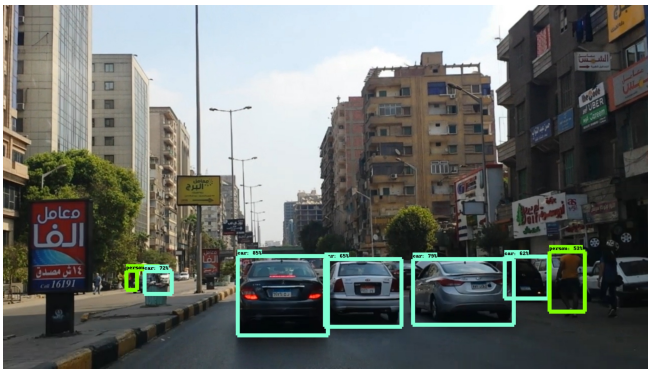

Fig. 7. The detection result from SSD with MobileNet architecture.

one frame in 3 seconds. So the speed margin is big for the SSD algorithm.

IV. CONCLUSION

In this paper, we have shown the accuracy for some of deep learning architectures on vehicle detection to use it with autonomous vehicles, if you want to make your detection task with fast timing and a good accuracy, MobileNetV2 can do the job since it is the fastest architecture among the others. If you need a good accuracy better than MobileNetV2 but needs more computation power with a moderate amount of classes, Inception-ResnetV2 or InceptionV3 can do this task. Finally, if you have a dataset with a huge amount of classes with more than 500 classes, NASNet can get this done. Also in meta-architectures algorithms, if you are willing to get high speed with acceptable accuracy, SSD can make a good detection with good speed. But if you are willing to a very high accuracy with a slow detection, Faster R-CNN can make this target perfectly.

REFERENCES

[1] A. M. Turing, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, 1950, 1.

[2] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of classifier methods: A case study in handwritten digit recognition," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition (Cat. No.94CH3440-5)*, vol. 2. IEEE Comput. Soc. Press, 1994, pp. 77–82, 2.

[3] S.-H. Lee, M. Bang, K.-H. Jung, and K. Yi, "An efficient selection of HOG feature for SVM classification of vehicle," in *2015 International Symposium on Consumer Electronics (ISCE)*. Madrid, Spain: IEEE, Jun. 2015, pp. 1–2, 3.

[4] T. Serre, L. Wolf, and T. Poggio, "Object Recognition with Features Inspired by Visual Cortex," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. San Diego, CA, USA: IEEE, 2005, pp. 994–1000, 4.

[5] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010, 5.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, 6.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778, 7.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *Proceedings of the ACM International Conference on Multimedia - MM '14*. Orlando, Florida, USA: ACM Press, 2014, pp. 675–678, 8.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, 9.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, vol. 9905, pp. 21–37, 10.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv:1506.02640 [cs]*, Jun. 2015, 11.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 2818–2826, 12.

[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *arXiv:1602.07261 [cs]*, Feb. 2016, 13.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861 [cs]*, Apr. 2017, 15.

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv:1801.04381 [cs]*, Jan. 2018, 15.

[16] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," *arXiv:1707.07012 [cs, stat]*, Jul. 2017, 16.

[17] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive Neural Architecture Search," *arXiv:1712.00559 [cs, stat]*, Dec. 2017, 17.