Evaluating Image Classification Models for FPGA Board Status Detection

Misha Baitemirova
Stanford University
medinab@stanford.edu

Abstract

In this project, we evaluated 2 deep learning models for image classification: VGG-16 and EfficientNetB4 for automating field-programmable gate array, or FPGA, board light status detection of LabsLand boards by Intel. Data augmentation techniques were employed to improve the performance of both models, and CenterCrop was the method that was the least successful. Grad-CAM visualizations for both models show that EfficientNetB4, in addition to including important elements, was considering spurious features as important. Overall, VGG-16 performed significantly better than EfficientNetB4.

1. Introduction

Our aim for this project was to employ deep learning for computer vision techniques to automate field-programmable gate array, or FPGA, board light status detection of LabsLand boards by Intel. LabsLand is an education technology startup that provides remote access to physical laboratories located in educational institutions in different parts of the world. It provides real-time access to hardware, thereby, enabling universities that can't purchase certain types of laboratories, for financial reasons or otherwise, to offer courses that rely on such laboratories. [8] One of the many labs offered by LabsLand is an access to Intel FPGA boards located in several countries, including at the University of Washington in Seattle, USA, where students can develop code and test it using those FPGA boards in real-time. Anytime a code is pushed to the board, one or more light indicators located on the board are turned on. A persisting problem has been identifying when those light indicators are on and when they are off due to varying light conditions and due to different positions of those boards, that together constitute a challenging condition. This project aims to evaluate deep learning models for image classification for detecting the light indicator status on FPGA boards. Therefore, the input is an image of the FPGA board and we use a CNN to predict which out of 10 LED indicators is on.

2. Related Work

2.1. Multi-Class Classification

Supervised learning is a sub-field of machine learning and artificial intelligence that consists of using labeled data for training a model. There are two types of supervised learning: classification, when working with output variables that are discrete, and regression, when working with continuous output variables. A classification task in machine learning is performed in a self-supervised learning manner. When dealing with data that consists of two discrete classes and can be classified into 2 outcomes, the classification task is a binary classification. In binary classification, we only require one classifier and can visualize the results in a very effective manner using a confusion matrix. When the datasets consists of more than 2 discrete classes, the task is multi-class classification that predicts a single class label for each data point. This is different from multi-label classification, where one class can be assigned more than one class label. Multi-class classification requires more than one classifier: essentially, we are using binary classifies to differentiate between different pairs of data. There are two ways of constructing such pairs: one-versus-rest, or OvR, and one-versus-one, or OvO. In OvR, one classifier is trained per class, where data points of that class are treated as positive examples and all other classes are treated as negative examples. A potential issue with this method is class imbalance. Even if each class is balanced, in the OvR approach, we have a single class with positive examples and (n-1) times more negative examples for n number of classes. One potential way to address this is by oversampling (duplicating the samples in the minority class), for example by using a method called SMOTE, or Synthetic Minority Over-sampling Technique. [3] However, this can lead to over-fitting the minority class. In OvO, a pair of classes is created from the original dataset and a classifier is trained on that pair. OvR and OvO are illustrated in Figure 1. [5]

2.2. Data Augmentation

Traditional data augmentation techniques include affine image transformations, including shearing, color modifica-

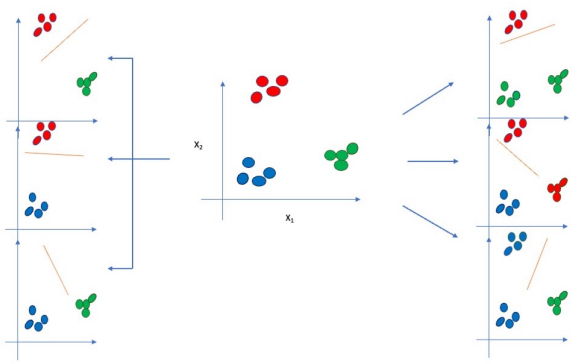

Figure 1. One-versus-One, on the left, and One-versus-Rest, on the right

tions, rotation, refection, and scaling. One paper found that classical data augmentation techniques were one of the most successful strategies for an improved performance on an ImageNet subset when evaluating data augmentation techniques and restricting the dataset to only 2 classes. [9] While dogs versus cats classification task is substantially different than detecting the state of FPGA boards via light detectors, data augmentation is a reasonable strategy to employ. It is important to note that not all data augmentation techniques can be generalized. For example, horizontal flipping, in some medical images and in our case, is not a label preserving data augmentation method. [11] Another team analyzed the effect of traditional data augmentation methods and found that it significantly improved model's performance, however, they did not evaluate the impact of each data augmentation method individually, which could be useful in our case. [7]

2.3. Deep Transfer Learning (DTL):

Classical machine learning techniques require manual feature design, which can be a significant disadvantage. In contrast, deep learning techniques learn and extract features through supervised and semi-supervised learning. The task of understanding and identifying patterns is difficult, therefore, deep learning models require a substantial amount of training data. In some fields, such as medicine, the amount of data available is not sufficient for proper training and learning. That is when transfer deep learning is useful: a model is not trained from scratch, but rather is pretrained on a large body of data. One example is VGG-16 that is pretrained on 1.28 million images of 1000 object categories, the ImageNet Large Scale Visual Recognition Challenge of 2014. [4]. In their paper on the *Differentiation of Active Corneal Infections from Healed Scars Using Deep Learning*, authors used VGG-16 to classify eye images acquired from 2 hospitals into healed scars and active infections. [14] It can be a lengthy process to set up clinical trials to generate enough data for training a deep learning model from scratch, therefore, in certain applications, such as eye infections, transfer deep learning can be the best option. However, there can be issues due to the mismatch in domains and input dimensions, leading to decreased accuracy and over-fitting. One way to address the mismatch in domain could be fine-tuning the feature learning layers of the pretrained model. [10] Another paper evaluated common state-of-the-art deep learning models for detecting COVID-19 using chest CT scans. One of the models they evaluated was VGG- 16 that was pretrained on ImageNet. They found that using pretrained weights resulted in good model performance, even if the CT scans were not part of the original training distribution.[4] The paper shows a fast and practical method using pretrained models. However, a limitation of this approach is that the sensitivity of the best performing model was 77.66%, which is low for healthcare. And while they addressed the class imbalance in their evaluation metrics, they did not perform data augmentation to address class imbalance before the training. [7]

3. Methods

3.1. Models

The baseline model evaluated for this project was VGG-16 against EfficientNetB4. VGG-16 is a larger model with more than 7x parameters than EfficientNetB4. For this project, VGG-16 was selected as the state-of-the-art model and EfficientNetB4 for the purposes of evaluating a smaller model and improving its accuracy using data augmentation techniques. Details on both models are in Figure 2.

Model	Size (MB)	Top-1 accuracy	Parameters	Depth
VGG-16	549	71.3%	138.4M	16
EfficientNetB4	75	82.9%	19.5M	258

Figure 2. VGG-16 versus EfficientNetB4

3.1.1 VGG-16

In 2014, VGG-16, a model presented by Karen Simonyan and Andrew Zisserman won the 1st and 2nd places in object detection and classification in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)of 2014. It has 134,305,611 parameters, 16 layers (13 convolutional, 3 fully connected). It was pretrained on ImageNet with an input size of 224x224x3 but it can also work with 256x256x3 images. Convolutional layers are followed by a rectified linear activation unit (ReLU), which is an activation function that outputs the input value directly when it is more than

zero, and returns zero otherwise. There are 5 max pooling layers that down-sample the input.[7] The output node uses softmax function that converts the output of the final layer into a vector of probabilities that all sum to 1. [12]

Its architecture is illustrated in Figure 3.

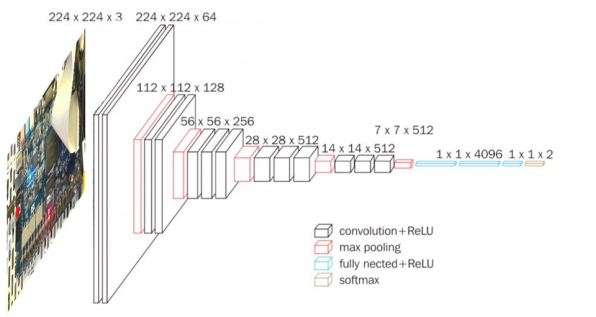

Figure 3. VGG-16 architecture

3.1.2 EfficientNet

EfficientNet represents a family of models presented by Google AI in 2019. They have less parameters compared to most state-of-the-art models, such as VGG-16 (19.5 million parameters in EfficientNetB4 versus 138 million in VGG-16). The architecture common to all the models in the family is illustrated in Appendices. EfficientNet proposes a new scaling method that uniformly scales all dimensions of depth, width, and resolution using a compound coefficient, which means that depending on the size of the input image, the network needs more or less layers to increase the receptive field and more or less channels to capture more fine-grained patterns. [13] The architecture common to all models in the family and the types of scaling are illustrated in Figures 4 and 5.

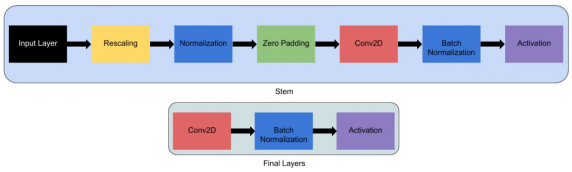

Figure 4. EfficientNet architecture common to all models in the family

There are models that have comparable sizes, such as ResNet-18 and -34. [6] And there are models that use scaling, for example, width scaling, such as MobileNet. However, instead of only scaling in one dimensions, EfficientNet models scale width, depth, and input size, which results in higher capacity to learn.

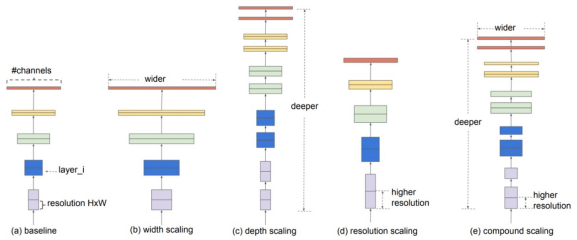

Figure 5. EfficientNet scaling

3.2. Evaluation Metric

In order to quantify model's performance, we will be using the accuracy metric, which measures how close the predicted labels are to ground truth labels. The classes are balanced, therefore, accuracy is an appropriate metric to use. The formula for accuracy is illustrated below:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

In addition, we used confusion matrices as they provide with a direct illustration of correct and incorrect classifications.

3.3. Saliency Map

We used Gradient weighted Class Activation Map, or Grad-CAM, to visualize areas of an input image that the model found important when making the classification decision. It is class-discriminative and works by examining the gradient information flowing into the final convolutional layer. The output is a saliency map for the class label that is being analyzed. Details on Grad-CAM are illustrated in Figure 6.

4. Dataset and Features

Data was generated by capturing photos from a video stream at three different locations at different times of the day to capture varied light conditions. The purpose was to generate images with each of the 10 LED lights on, and images with all of the LED lights off, resulting in 11 classes for the classification task. Examples are illustrated in Figures 6 and 7. Number of images per location and the number of images used for training/validation/test are summarized in Figure 8.

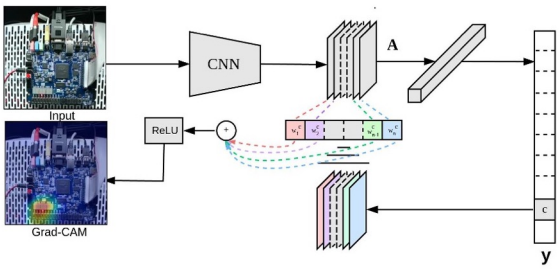

Figure 6. How Grad-CAM works

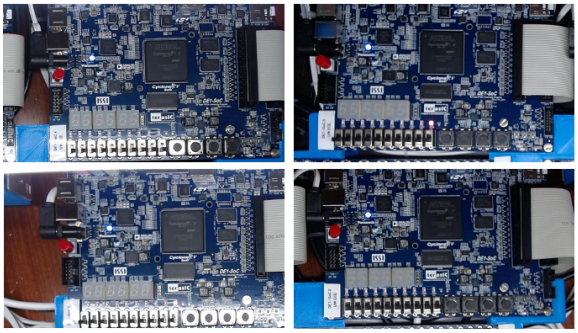

Figure 7. Example photographs of FPGA boards

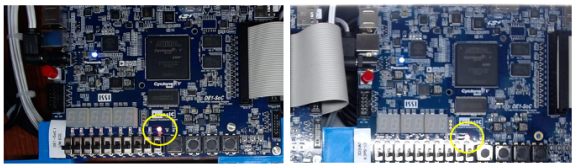

Figure 8. Example photographs of LED buttons that are turned on, under two different light conditions and locations (one board is positioned lower than the other)

Location	Number of boards	Total number of images	Training	Validation	Test
Pamplona, Spain	9	3960	2376	792	792
Pamplona, Spain	6	2640	1584	528	528
Seattle, USA	7	3080	1848	616	616

Figure 9. Summary of data statistics per location)

VGG-16 was pretrained on ImageNet and the input sizes were 224x224. Since our images ranged from 540x960 to 720x960 in size, they were resized to match VGG-16's original input size. Moreover, VGG-16 expected a normalized input, therefore, we applied normalization as well. We used the same mean and standard deviation used for ImageNet.

EfficientNetB4, also pretrained on ImageNet, required an input size of 380x390, therefore, the original images were resized to the proper size. We also used the same mean and standard deviation used for ImageNet.

We have performed data augmentation on input images to increase the training set size and regularize the network to avoid overfitting. More specifically, we employed affine data augmentation techniques to proxy varying positions of boards and varying camera angles.

5. Experiments/Results/Discussion

5.1. Model Setup

Due to the multi-class nature of the dataset, the experiments focused on evaluating multi-class approaches. Most deep learning models employ one-versus-rest, or OvR, approaches. In addition, OvR is an extension to the binary classification. As a result, this is the multi-class classification that was tested for the purposes of this project by evaluating models VGG-16 and EfficientNetB7. We used early stopping for both models and used default model settings, such as a learning rate of 1e-5,cross-entropy loss function, batch size of 40, and an Adam optimizer, which is a replacement optimization algorithm for stochastic gradient descent in deep learning, for VGG-16. For EfficientNetB4, we used default settings as well: learning rate of 1e-3 and an Adam optimizer.

5.2. Results

5.2.1 VGG-16 Results

The confusion matrix and Grad-CAM visualization for VGG-16 are illustrated in Figure 10 and 11.

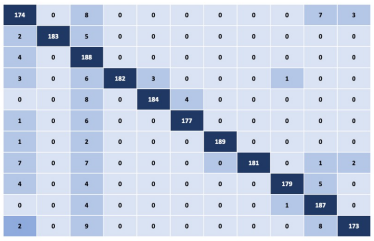

Figure 10. Confusion matrix for the best result of VGG-16

5.2.2 EfficientNetB4 Results

The confusion matrix and Grad-CAM visualization for EfficientNetB4 are illustrated in Figure 12 and 13.

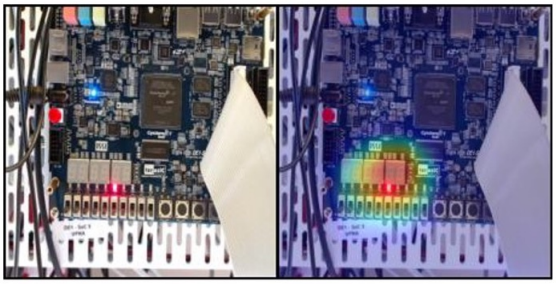

Figure 11. Confusion matrix for the best result of VGG-16

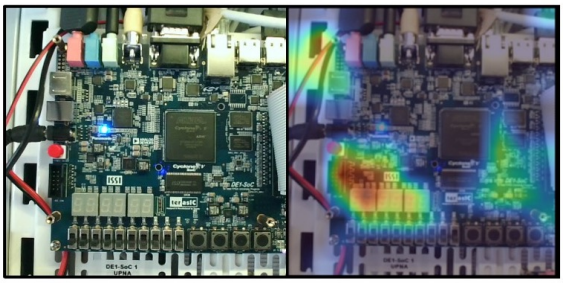

Figure 12. Confusion matrix for the best result of EfficientNetB4

5.2.3 VGG-16 versus EfficientNetB4

Data augmentation methods that were tested and their accuracy results are summarized in Figure 9.

Augmentation method	VGG-16	EfficientNetB7
None	93.43%	39.97%
Center crop	85.79%	24.81%
Random resize	89.75%	50.76%
Random rotation	94.29%	34.89%
Random color change	93.52%	26.81%
All	88.81%	34.89%

Figure 13. VGG-16 and EfficientNetB4 results

Both models' Grad-CAM visualizations are illustrated in Figures 12 and 13. We can see that both VGG-16 and EfficientNetB4 were considering the LED indicators as important. However, EfficientNetB4 was including spurious features, such as board cables that never move.

5.3. Discussion

Overall, VGG-16's performance was significantly better than the performance of EfficientNetB4. The performance of VGG-16 without no classical data augmentation techniques was comparable to the best performing model using data augmentation. One reason that could explain that is the data capturing process: our initial dataset taken at different times of the day and different locations already incorporated rotation and color invariance and had an adequate number of training examples.

Random Rotation with an angle of 15◦ and Random Color Change led to the best VGG-16 performance. Randomly rotating each image mimics the camera angles and positions, and as a result, the model generalizes well to a variety of angles. In practice, at every university where FPGA boards are installed, camera angles don't have a lot of variation, therefore, the angle of rotation we chose was reasonable. Randomly adjusting the brightness, contrast, and saturation mimics differences in images due to different cameras and different conditions, such as day versus night. Therefore, in addition to increasing the training dataset size, these data augmentation techniques also incorporated rotation and color invariance into the model. Center Crop performed worse than others. One reason could be that this data augmentation technique was not a label preserving transformation. We resized each image to 256x256 before cropping to 224x224. However, when analyzing FPGA board images we realized that some images had the light indicators at the edge of the image. Center Crop in those images would essentially be cropping out all of the LED indicators. Lastly, there was not a significant variation between training, validation, and test accuracy values for VGG-16, which indicates that model did not overfit to the training dataset but further evaluation is needed.

There are several advantages to using smaller models, such as EfficientNetB4. They require less computational resources and less training time, which makes it more accessible to a larger number of users and deep learning practitioners. However, it performed poorly on our dataset. It novel scaling method, that includes scaling in width, depth, and input size leads to a smaller model with higher learning capacity. However, in practice, the good performance of EfficientNet as described in the original paper, has not been reproduced as expected. One reason could be the nature of EfficientNet models. GPU hardware accelerators are intended and designed for larger models, and therefore, can have an impact on the performance of EfficientNetB4. [1]

FLOPs, or floating-point operations, is a way to measure computational resource requirements. [2] In the case of EfficientNet models, we are trying to balance between adding more FLOPs and improving the model performance but it has a disadvantage: it increases the amount of data that flows from one layer to another. GPUs are designed for models with less movement of data between layers, and as a result, this can lead to a poorer performance. Lastly, the reason could be the hyperparameters we used for this project and further hyperparameter fine-tuning and engineering can improve model's performance.

Since both models were pretrained on the same dataset, domain shift is unlikely to be the reason for the difference in model performance.

6. Conclusion/Future Work

For images with complex features, the model with the bigger size and higher number of parameters performed better compared to the smaller model. Advantages of larger models include larger learning potential but it can also lead to over-fitting. Advantages of smaller models include less demanding computational requirements but due to a smaller number of parameters, these models can lead to poorer expected performance. For the purpose of FPGA board status detection, smaller models would be ideal, therefore, the next steps would be testing deep learning data augmentation methods and hyperparameter tuning to improve the performance of EfficientNetB4. Another potential solution would be testing other small image classification models.

References

[1] Gpu for deep learning. 5

[2] Flops, May 2022. 6

[3] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority oversampling technique. *CoRR*, abs/1106.1813, 2011. 1

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2

[5] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Sola, and Francisco Herrera. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, 44:1761–1776, 08 2011. 1

[6] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020. 3

[7] Mohamed Loey, Florentin Smarandache, Nour Eldeen, and M Khalifa. A deep transfer learning model with classical data augmentation and cgan to detect covid-19 from chest ct radiography digital images. 08 2021. 2

[8] Luis. Labsland fpga lab instances deployment at unifesp (brazil), Feb 2020. 1

[9] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017. 2

[10] Huy Phan, Oliver Y. Chén, Philipp Koch, Alfred Mertins, and Maarten De Vos. Deep transfer learning for single-channel automatic sleep staging with channel mismatch. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019. 2

[11] Connor Shorten and Taghi Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 07 2019. 2

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 3

[13] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. 3

[14] Mo Tiwari, Chris Piech, Medina Baitemirova, Namperumalsamy Prajna, Muthiah Srinivasan, Prajna Lalitha, Natacha Villegas, Niranjan Balachandar, Janice Chua, Travis Redd, Thomas Lietman, Sebastian Thrun, and Charles Lin. Differentiation of active corneal infections from healed scars using deep learning. *Ophthalmology*, 129, 08 2021. 2