APPLICATION OF TRANSFER LEARNING & AUTOMATED MACHINE LEARNING IN IMAGE CLASSIFIERS FOR DOG BREEDS

Brandon Attai, Tahsin Chowdhury, Kelten Falez, Timothy Mok, Aron Saengchan, and Yong Jun Zhu

Department of Electrical & Software Engineering, Schulich School of Engineering,
University of Calgary, 2500 University Dr. NW, Calgary, AB, T2N 1N4

ABSTRACT

Convolutional neural networks (CNN) have proven to be one of the most valuable tools for image analysis and classification tasks. Building upon a shared-weight architecture known as feature maps, CNN models can sometimes require significant time and effort to train and tune properly in providing useful results. This study reports the results of training and applying two popular CNN models in the classification task of a collection of dog breed images – ResNet50 and VGG16. A series of systematic tests were performed to optimize and achieve higher validation accuracies. To further extend these methods, the best iterations were ensembled together by capturing the feature maps of each model. The results of these CNN models were also tested against automated machine learning methods, performed using the AutoKeras and AutoGluon framework, with the benefits and drawback of each being evaluated. Gradient-weighted Class Activation Mapping was applied to visualize how select models progressively analyzed the provided input images in arriving to generalized predictions.

***Index Terms* —** Image classification, convolutional neural networks, ensemble learning, AutoML, Grad-CAM

GitHub repository: https://github.com/acsaeng/enel645-group17

1. INTRODUCTION

Deep computer vision has made great advancements by introducing two additional fundamental building blocks – convolutional and pooling layers. Pioneered by the LeNet-5 architecture, this convolutional neural network (CNN) built upon the fundamentals of fully connected layers and sigmoid activation functions [1]. The convolutional layers that encompass a CNN offer the benefit of not connecting to all the pixels in the input image but only those in their receptive field. This allows the network to focus on smaller features in the first hidden layer, then subsequent layers assemble into higher level features as the network progresses [2].

While there has been significant research in the field of deep learning, with the theoretical understanding of a CNNs well researched and documented, the field of deep learning still proves challenging due to its inherent complexity [3]. Some of the common challenges of training and implementing CNNs effectively are the dependencies on the large amounts of data required, imbalanced datasets, overfitting, and computational requirements, among others. However, despite the ability to achieve accurate results, one of the most pressing issues of CNNs are its inability to perform well when presented with non-idealistic images from real world scenarios, which can typically be solved with more human intervention and providing more training data [4].

Nonetheless, solutions have been derived to solve some of the common issues faced by deep learning researchers such as transfer learning and automated machine learning (AutoML). Transfer learning provides the benefit of being able to apply a pre-trained model to a closely related task, thereby reducing the amount of data and computational resources required for training [2]. Moreover, when applied in a deep learning context, transfer learning has multiple strategies such as using pre-trained models as feature extractors or fine-tuning the layers in pre-trained model to improve its effectiveness, depending on the application and its relative closeness to the task being addressed [5]. Likewise, AutoML aims to make machine learning accessible to non-machine learning practitioners by providing an automated tool that can perform actions such as data preprocessing, feature exploration, hyperparameter optimization, and model and architecture searching [6].

This study presents the multiclass image classification problem of identifying dog breeds using a series of different deep learning methods. Numerous models were developed in order to correctly classify a dog's breed, provided a collection of input images. Popular convolutional neural network models, including the ResNet50 and VGG16 models were applied, using transfer learning methods to ensure the models adhered to the problem. Other methods aimed to use AutoML (AutoKeras and AutoGluon) in attempting to find other means of successfully classifying the data. Lastly, gradient-weighted class activation mapping (Grad-CAM) was utilized to visualize important aspects of the input data used by the best model. The results highlight the usefulness of CNNs and AutoML in image classification problems, especially when the input data is shown to be fairly complex.

2. RELATED WORK

Historically, studies have taken a variety of approaches in solving multiclass image classification problems similar to the one proposed here. Shi et al. applied four different pre-trained models – ResNet18, DenseNet161, AlexNet, and VGG16, on the same Kaggle dog breed dataset used in this study [7]. The loss and accuracy were explored based on the applying these pre-trained models with two different optimizers – Adam and SGD – with DenseNet161 ultimately performing the best.

Ráduly et al. explored the dog breed identification problem based on the Sanford Dogs Dataset [8]. Their approach involved applying the NASNet-A mobile architecture and the Inception ResNet V2 network, with the latter performing much better due to its deeper network. A Grad-CAM analysis applied by this research group found that the heads of the dogs were the primary focus when the models attempted to classify a dog's breed.

When examining classification problems, Fast and Jensen discovered that stacking multiple models could reduce the amount of bias and allowed the combined model to perform better [9]. Likewise, Chand et al. also found that by stacking an already strong classifier that achieved a 91% accuracy with other machine learning models, the combined model's ability to detect cyber-attacks was increased by 6% [10].

By applying the principles that other researchers have come across, this project aims to further explore the findings on the multiclass image classification of dog breeds while affirming and building upon the potential improvements to pre-trained models.

3. MATERIALS & METHODOLOGY

3.1. Dataset

The 10,222 images containing different dog breeds were divided into training and validation sets using a 0.8:0.2 ratio. This resulted in a training set containing 8178 images and a validation set containing 2044 images, each belonging to a total of 120 classes. The entire dataset was sourced from a previous Kaggle competition [11]. A set of select sample images from the training dataset are provided in Fig. 1. Before developing the forthcoming CNN models, this training data was preprocessed using the preprocess input feature from Keras. Various tools were used to visualize and describe the elements in the input dataset.

Unfortunately, there were no labelled test sets that could be used to evaluate the final model performance. Therefore, validation accuracy was the primary metric used to gauge the performance of the final models. However, as an alternative, a small test set of 13 images encompassing 12 different classes was compiled from Google Images in order to directly test the models.

Fig. 1. Sample images from the Kaggle Dog Breed Identification Dataset

3.2. CNN Models

3.2.1. ResNet50 & VGG16 Models

Two common CNN models – ResNet50 and VGG16 – were selected as the primary architectures employed for this dataset. ResNet50 was chosen because it utilizes skip connections to mitigate vanishing gradients. Likewise, VGG16 was selected because its small size convolution filters allow it to have more weight layers, leading to an improved performance. Both these models also draw from ImageNet, using its weights as part of its architecture. The layers were initially frozen, and the fully connected section of the pre-trained model was replaced with a custom output layer. This layer consisted of various dense and dropout layers, as well as utilizing a softmax activation function for all the 120 dog breed classes.

3.2.2. Model Training

During training, a systematic approach was undertaken, whereby each model was progressively tuned to improve the training and validation score of each subsequent iteration. The specifications on Table 1 were implemented for every training iteration of both the VGG16 and ResNet50 model architectures.

Data augmentation was performed utilizing the Keras image data generator feature. This feature generated batches of tensor image data with real-time data augmentation. A horizontal flip technique was selected along with a batch size of 32 and an image size of 224 × 224 pixels. The generated training and validation images were shuffled as sparse, 1D NumPy arrays of integer labels.

After each model was trained, it was re-trained again with the intent of fine tuning it by unfreezing a fraction of the layers that were non-trainable in the first round. The best model was saved after each epoch. All models we trained on Google Colab notebooks using a Tesla P100 GPU.

Table 1. ResNet50 and VGG16 model specifications.

Specification	Value
Activation function	Softmax
Loss function	Categorical cross entropy
Evaluation metric	Accuracy score
Learning rate	0.0001 (decayed by factor of 2 per 10 epochs)
Number of Epochs	20
Early stopping	True
Patience	10

3.2.3. Hyperparameter Tuning

Several hyperparameters were tuned during the iterative training process of the ResNet50 and VGG16 models, including the batch size, the image size, and image augmentation (rotation, width shift, height shift, shear range, zoom range, horizontal flip, and vertical flip) parameters. Other features like the batch number and image size of the models were also considered.

In terms of model architecture, several different configurations of the fully connected portion of the ResNet50 and VGG16 models were attempted. This included the application of batch normalization and 2D global average pooling layers. These conditions were extensively tested with varying amounts of dense layers and dropout layers with different probabilities, as well as several different optimizers, such as RMSprop, Adam, and stochastic gradient descent.

3.3. Ensemble Model

An attempt to further improve the validation accuracy was conducted employing ensemble learning methods, incorporating both the developed ResNet50 and VGG16 models (Fig. 2). To successfully perform this, a method was adapted from [12], whereby all the features were extracted from each optimized model and an input layer with a dropout of 0.7 was added.

The same specifications of a softmax activation function, categorical cross entropy loss function, and an accuracy score metric from the previous models were also defined for this ensemble model. A detailed setup of the ensemble is further illustrated on Figure 2. By combining these feature maps, the objective was to acquire a more diverse range of features in order to classify the dog breed images more accurately.

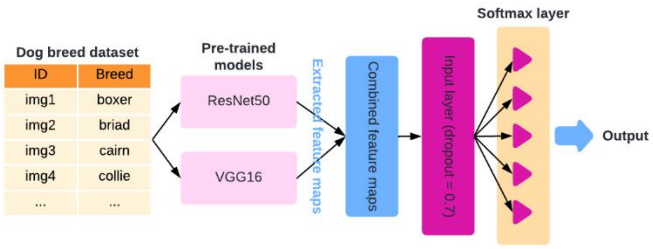

Fig. 2. Architecture of the ensemble model.

3.4. AutoML

Two different options for AutoML were explored in this study – AutoKeras and AutoGluon – with the intent of comparing the benefits and drawbacks of both.

3.4.1. AutoKeras

The open-source library AutoKeras was the first option used for performing AutoML on this deep learning classification task. AutoKeras builds complex and effective deep learning models by utilizing AutoML frameworks.

For this labelled dataset, both ResNet and Xception models were explored. It must be noted that the maximum trials argument (which is the maximum number of different Keras models explored) was set to 100. Prior to running AutoKeras, the training and validation data was organized into separate directories, where each was named after one of the 120 classes and only contained images unique to that class.

3.4.1. AutoGluon

For image classification tasks, the AutoGluon library provides a simple fit function that trains different neural network configurations, including different architectures such as ResNet50, VGG16, EfficientNet and a range of different hyperparameter values for hyperparameters such as those previously described.

Training data was inputted into AutoGluon's image predictor in the form of a .csv file containing two columns – the path to each image in the training set and the corresponding label. Each model was trained for a time limit of 2 hours and 100 epochs.

3.4. Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) uses the gradients of any target flowing into the final convolutional layer to produce a localization map highlighting the important regions in the image for predicting the concept, thus making the model more interpretable [13]. To understand how the best performing ResNet50 model successfully classified an image to its correct class, a test image was used to capture heatmaps for different convolutional layers in the architecture.

4. RESULTS & DISCUSSION

4.1. Model Performance

Table 2 and Fig. 3 summarize and compare the performance of each tested model. This includes the baseline ResNet50 and VGG16 models with all the pre-trained layers frozen and having only one dense layer as the output, as well the best iterations of these model after manual

hyperparameter tuning. It also presents the metrics of the best models from AutoKeras and AutoGluon. The architecture of these models is discussed more in detail in the following sections.

Table 2. Training accuracy, validation accuracy, and train of each model.

Model	Training accuracy (%)	Validation accuracy (%)	Training time (s)
Baseline ResNet50	99.9	62.1	800
Best ResNet50	95.4	79.8	893
Baseline VGG16	99.9	53.1	808
Best VGG16	65.7	68.7	2034
Ensemble	42.6	89.3	14
AutoKeras	97.5	82.9	2604
AutoGluon	85.5	89.9	2083

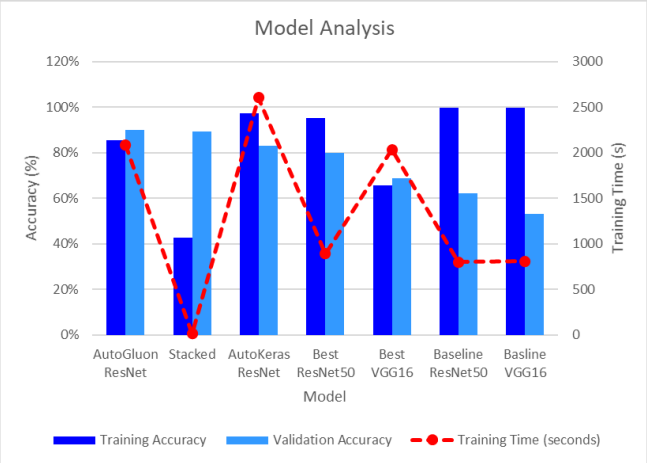

Fig. 3. Plot comparing the training accuracy, validation accuracy, and training time of each tested model.

The model architectures for the best ResNet50 and VGG16 used in this study are shown in Table 3 and Table **4**, respectively. An iterative process was employed to optimize the hyperparameter in attaining higher overall validation accuracies.

Even after several iterations of manual hyperparameter tuning, the best models with ResNet50 and VGG16 architectures achieved validation accuracies of 79.8% and 68.7%, respectively. One explanation for this is the relatively low number of observations the dataset used per class. A rule of thumb for getting good performance from deep learning models is to have at least 5000 observations per class [14]. In this experiment, each class in this dataset only has a handful of observations, with the most having only 120.

Another explanation could be the size of the dataset itself, as deep learning is an extremely data-driven process, especially in regard to performance [15]. Model capacity increases as the number of training samples increase exponentially. The number of training samples in the dataset used in this project was only just over 10,000 and may have contributed to the relatively low accuracy scores attained.

Table 3. Model architecture for the best ResNet50 model.

Layer (type)	Output shape	Param #
Input layer	(None, 224, 224, 3)	0
Slicing op lambda	(None, 224, 224, 3)	0
TF op lambda	(None, 224, 224, 3)	0
Functional (ResNet50)	(None, 7, 7, 2048)	23,587,712
Batch normalization	(None, 7, 7, 2048)	8,192
Global average pooling 2D	(None, 2048)	0
Dropout	(None, 2048)	0
Dense	(None, 2048)	4,196,352
Dropout	(None, 2048)	0
Dense	(None, 120)	245,880
Total params: 28,038,136 Trainable params: 26,530,936 Non-trainable params: 1,507,200		

Table 4. Model architecture for the best VGG16 model.

Layer (type)	Output shape	Param #
VGG16 (functional)	(None, 7, 7, 512)	14,714,688
Batch normalization	(None, 7, 7, 512)	2,048
Global average pooling 2D	(None, 512)	0
Dropout 1	(None, 512)	0
Dense 1	(None, 1024)	525,312
Dropout 2	(None, 1024)	0
Dense 2	(None, 256)	262,400
Dropout 3	(None, 256)	0
Dense 3	(None, 120)	30,840
Total params: 15,535,288 Trainable params: 819,576 Non-trainable params: 14,715,712		

4.1.1. Batch Normalization & Global Average Pooling

Batch normalization was implemented to accelerate the model training process. This was performed by standardizing layer inputs on a batch basis, resulting in the reduction in epochs for the network. Batch normalization allowed for experimentation with high learning rates and reduced the dependency for dropout [16]. Additionally, batch learning provided regularization to assist with mitigating error due to generalization.

A global average pooling layer was used in lieu of a flatten layer before the dense layers in the neural network. Global average pooling layers can be considered as a structural regularizer, as it can reduce overfitting. It does so by reducing the number of trainable parameters [17]. A global average pooling layer averages all the values according to the last axis in the tensor, whereas a flatten layer simply flattens the tensor by reshaping, resulting in a higher number of trainable parameters.

4.1.2. Ensemble Model Generalization

After tuning the hyperparameters for each individual model, the best validation accuracies achieved by the best ResNet50 and VGG16 models still left much room for improvement. By stacking these two models in incorporating both of their feature maps, the ensemble model was able to reach a validation accuracy of 89.3%, higher than that achieved by each of the models individually. It should be noted that the training accuracy was observed to be significantly lower than the validation accuracy due to the increased training loss as a result of the added dropout layers [18].

Because the ResNet50 and VGG16 models were trained individually before extracting the features and applied to the softmax layer, the stacked model was able to minimize its biases and generalize better to the new data [9]. This method focused on minimizing the weaknesses of separate models while focusing on their strengths so that the accuracy could greatly improve. The time required for the ensemble model to train (14 s) is significantly smaller than that of the individual models, although it still requires more time to compile everything together. However, it was concluded that the increase in accuracy still outweighs the many drawbacks in this classification task.

4.1.2. AutoKeras vs. AutoGluon

Upon applying both AutoML frameworks, there were many noticeable benefits of AutoGluon in particular. The first benefit of AutoGluon was its execution time, as it performed significantly faster than AutoKeras, requiring an average fit time of 1 h or 3600 s. The execution time of the AutoML tools used in this study was incredibly important given some of the time limitations. Further reports have concluded that AutoGluon also yielded the best average predictive score and the lowest computational effort for multi-class classification problems [19].

One significant drawback associated with AutoKeras was that it did not support cross-validation during the fitting phase. This drawback was addressed by utilizing a less-complex holdout training and validation set split to select and fit the models. This resulted in a longer execution time for AutoKeras compared to AutoGluon.

4.2. Optimization Strategies

4.2.1. Addressing Overfitting & Underfitting

When training the neural networks using transfer learning techniques for the ResNet50 model with ImageNet weights, one challenge that was encountered was underfitting. This is demonstrated when the validation accuracy exceeds the training accuracy and occurs when the model is not complex enough. Unfortunately, adding additional training data in this case was not a viable solution. To alleviate this issue, an additional fully connected dense layer was added to increase the complexity of the model along with another dropout layer.

Conversely, overfitting is another common occurrence in neural networks, occurring when the model fits the training data too well, resulting in the training accuracy exceeding the validation accuracy. The objective here was to generate a model that learns the training data, while still generalizing well enough on unseen data in the validation set throughout the training process. A variety of options were considered in order to mitigate overfitting during hyperparameter optimization, some of which incorporated:

- Data augmentation
- Layer manipulation (freezing and unfreezing layers)
- Dropout layers
- Pooling layers
- Early stopping
- Regularization

It must be noted that not all methods implemented resulted in a successful reduction of overfitting, with some proving to have more effective results than others, such as adding a dropout layer after a dense layer and applying it to the transfer learning model.

4.2.2. Training Unfrozen Layers

As various pretrained models were implemented using transfer learning methods, namely ResNet50 and VGG16, one strategy to improve classification accuracy as well as reducing computational requirements involves freezing and unfreezing of trainable layers in the convolutional network. Previous research has shown that while there are several techniques to control overfitting, such as dropout and regularization, these methods are best suited for conventional fully connected networks.

By freezing and unfreezing layers, it allows the pre-trained weights to be kept during initial training by freezing the layers in the network, then subsequently unfreezing the lower convolutional layers for continued training. By unfreezing the lower layers, it builds on empirical evidence that different sections of a network learn different features, which may impact the outcome of the classification task. As

such, manual experimentation of the layers was used to find the optimal finetuning parameters.

4.3. Grad-CAM Analysis

Fig. 4 illustrates the Grad-CAM heatmaps for the convolutional layers in the best RestNet50 model, sequentially for an image that was predicted correctly. Some logic can be deduced here, as the heat maps are studied progressively. The first two convolutional layers seemed to detect the background of the image, while the subsequent two layers detected the contours of the dog's body shape. Finally, the last convolutional layer detected features that may be unique to the breed in question, such as the shape of its head. Through this process provided by Grad-CAM, a deeper understanding was acquired on how the convolutional layers were able to progressively distinguish certain features of the image in order to eventually arrive at a prediction.

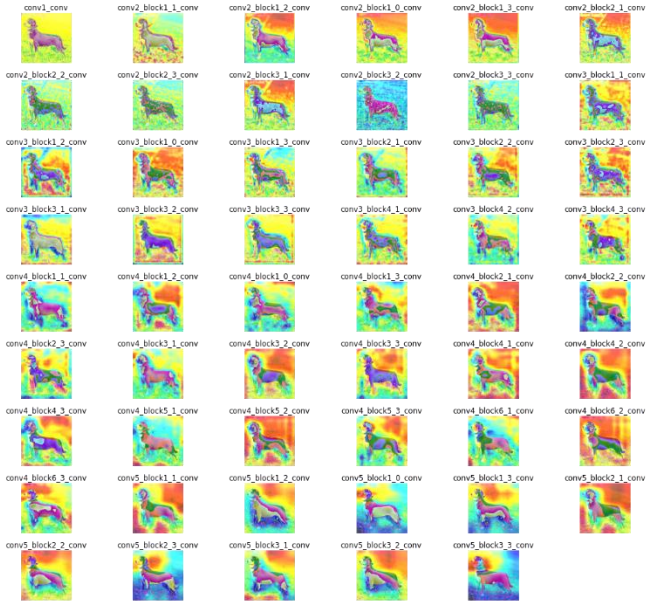

Fig. 4. Progressive Grad-CAM heatmaps of the convolutional layers from the best ResNet50 model.

5. CONCLUSIONS

In this study, two pre-trained CNN models – VGG16 and Resnet50 – with ImageNet weights were used to build multiclass image classifiers for a dataset of different dog breeds containing 120 different classes. These models were fine-tuned using different data augmentation techniques, hyperparameters, and architectures for the fully connected layers. The best models for the ResNet50 and VGG16 model achieved validation accuracies of 79.8% and 68.7%, respectively. These two models were further extended by stacking their respective feature using ensemble methods to improve the accuracy score, successfully doing so with a validation accuracy of 89.3%. Lastly, to fully interpret some of these results, a Grad-CAM analysis was performed on the best ResNet50 model.

The performance of transfer learning was then compared with AutoML methods. Two AutoML libraries – AutoGluon and AutoKeras – were explored. With a validation accuracy of 89.9%, AutoGluon attained the best results with a significantly faster performance between the two. Therefore, it was concluded that if AutoML were to be used for such a task, AutoGluon would be the superior choice in this case, as it achieved a better validation accuracy in the least amount of time, while also using less computational resources.

6. REFERENCES

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278 - 2324, 1998.

[2] A. Géron, "Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow," Sebastopol, O'Reilly Media, Inc., pp. 447-481, 2019.

[3] L. Alzubaidi, J. Zhang, A. Humaidi, A. AlDujaili, Y. Duan, O. AlShamma, J. Santamaría, M. Fadhel, M. AlAmidie and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 53, pp. 2-50, 2021.

[4] P. Ahmadvand, R. Ebrahimpour and P. Ahmadvand, "How Popular CNNs Perform in Real Applications of Face Recognition," in *24th Telecommunications Forum (TELFOR)*, 2016.

[5] C. Sharma, "Transfer Learning and its application in Computer Vision: A Review," 2022.

[6] S. K. K. Santu, M. Hassan, M. J. Smith, C. X. Zhai and K. Veeramachaneni, "AutoML to Date and Beyond: Challenges and Opportunities," Virtual Data Scientist, 2020.

[7] W. Shi, J. Chen, M. Liu, and F. Liu, "Dog Breed Identification," 2010.

[8] Z. Ráduly, C. Sulyok, Z. Vadászi and A. Zölde, "Dog Breed Identification Using Deep Learning", in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, 2018, pp. 271-276, doi: 10.1109/SISY.2018.8524715.

[9] A. Fast and D. Jensen, "Why Stacked Models Perform Effective Collective Classification," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 785-790, doi: 10.1109/ICDM.2008.126.

[10] N. Chand, P. Mishra, C. R. Krishna, E. S. Pilli and M. C. Govil, "A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection," in *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring)*, 2016, pp. 1-6, doi: 10.1109/ICACCA.2016.7578859.

[11] Dog Breed Identification Dataset, Kaggle, Sep. 2017. [Online]. Available: https://www.kaggle.com/competitions/dog-breed-identification/data.

[12] D. Pradhan, “Inception + Xception + NASNetLarge + InceptionRes,” Coder Zero, Bengaluru, India, 2021, Accessed: Mar 26, 2022. [Online]. Available: https://www.kaggle.com/code/deepakat002/inception-xception-nasnetlarge-inceptionres/notebook

[13] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization,” Oct. 2016, doi: 10.1007/s11263-019-01228-7.

[14] I. Goodfellow, Y. Bengio, and A.Courville, "Deep Learning," in *Deep Learning*, MIT Press, p. 34, 2016.

[15] T. Linjordet and K. Balog, "Impact of Training Dataset Size on Neural Answer," in *Advances in Information Retrieval*, vol. 11427, pp. 828-835, 2019.

[16] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, vol. 37, pp. 448-456, 2015.

[17] M. Lin, Q. Chen, and S. Yan, “Network In Network,” *Neural and Evolutionary Computing*, Dec. 2013.

[18] Keras, "Why is my training loss much higher than my testing loss?" Keras.io. https://keras.io/getting_started/faq/#why-is-my-training-loss-much-higher-than-my-testing-loss (accessed Mar. 27, 2022).

[19] L. Ferreira, A. Pilastri, C.M. Martins, P.M. Pires, and P. Cortez, "A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.