Computer Vision: Object Detection and Recognition

Nayan Arora   
*Faculty of Science and Technology*   
University of CanberraCanberra, Australia  
u3249907@uni.canberra.edu.au

*Abstract*— Object detection and scene understanding has seen immense improvements in the recent past with Tesla self-driving cars being at the forefront. In layman language a Tesla can use the real time information from its cameras and lidar sensors to interpret and detect objects and other cars in its surroundings using the ideologies of computer vision and machine learning techniques. This report covers a smaller subset of the theory in Computer Vision - to introduce us to the concept - by training multiple classifiers to understand and compare results. In this report we compare the training and validation results produced by SVMs and CNNs on the Caltech\_UCSD\_Birds\_200\_2011 dataset that comprises of over 200 classes of bird images. The results produced are highlighted along with supporting research on methods of improvements.

Keywords— Image Classification, Object Detection, AI, Boundary Boxes, Computer Vision, Classifiers, Machine Learning, SVM, HOG, Decision Trees, Random Forest, CNN, Transfer Learning.

# Introduction

In this report the major theory in Computer Vision is explored by understanding the principal techniques or the building blocks of complex models like Resnet. Based on the preliminary research conducted, it is understood that the most widely accepted method for training machine learning model (for image/video data) is using Convolutional Neural Networks (CNNs) because of their versatility, efficiency, and effectiveness.

All major industry level models like Inception, VGG16, Alexnet, YOLO and others use the CNN architecture. But it is necessary to understand how an algorithm can train itself to make future predictions. It is important to learn how an algorithm can find and learn patterns. It is also necessary to know what parameters are being used by an algorithm and how they might influence the results produced. These are the areas explored through this report.

## Motivation

I am a young professional and have seen technology advance from using a landline to using a wristwatch to make phone calls. We have seen significant advances in the AI spectrum and the world is only moving forward. Thus, the real motivation is the urge to understand AI and the theory in ML.

# Litearure Review

## Workshop Notes

Extensive discussions in our workshop and reviewed notes on methodologies from every workshop over the 12 weeks.

## Computer Lab Notes

Reviewed all the sample code provided during the lab hours. Also, it is worth mentioning that a lot of the code from our labs was reused and appropriately referenced in producing this report.

## Online research

Multiple research papers [1], [2], [3], [4] were used for understanding pre-existing solutions in the domain. It was fascinating to read through these papers and learn the different approaches that were used in tackling the 200-class bird image dataset.

One thing found common in all the research papers was the extensive use of CNN based algorithms that produced highly accurate results.

# Methodology

## Classic Machine Learning Approach

In my research I investigated the following three feature extraction methods:

* Histogram of Oriented Gradients (HOG): They help capture edge information that is crucial in outlining shapes [5]. They work particularly well for object detection problems.
* Scale-Invariant Feature Transformation (SIFT): Helps in identifying and describing the invariant local features [5], that can be used for matching different objects in various scenes and backgrounds, thus a useful feature extraction tool for image data.
* Speeded-up Robust Features (SURF): This works like how SIFT works but is relatively faster, thus can be used for real-time object detection projects.

The above listed feature extraction methods are combined with an actual classifier or algorithm that helps in learning patterns and information in data that can be then used for making predictions. The classifiers researched for this project are:

* Logistic Regression: This linear regressor has been widely implemented for classification tasks because of its ease of interpretability. The LR uses a logistic function to estimate the probabilities of dependent and independent variables.
* Random Forest: This ensemble learning based classifier uses a combination of decision trees for learning and is known for generalizing well, thus being robust to overfitting.
* K-Nearest Neighbor: Effective learner of local patters as it does not make any assumptions. It only learns what it sees, no estimations. Thus, a good choice for supervised learning.
* Support Vector Machine: It finds the hyperplane that maximizes the margin such that the two hyperplanes are closest to the data points. In this way it can learn and classify data.

For this project I used the 20-class development dataset to try multiple different combination of the above listed classifier and feature extraction methods to produce an overall average prediction accuracy. I got the best results using the Support Vector Machine (SVM) with HOG feature extraction method.

I used a similar trial and error method to also understand the different parameters available for optimization.

In SVM’s the criterion of the function that is being optimized is the margin. The margin basically refers to the distance between two parallel decision boundaries or support vectors. So, the SVM models basically tried to find the best hyperplane that maximizes the margin such that the two support vectors fall closest to the data points [6]. Either side of the data point is the predicted binary class. The goal is to have the widest possible margin as the width shows the confidence level of or the accuracy level of the model [6].

For the problem at hand, we have multiple classes. So, we basically fine tune multiple binary classifiers working in parallel to form a multi-class classifier. Below is a brief explanation on the parameters used:

* Kernel Function: Of the three options, that is, Radial basis function (rbf), Polynomial Function and Linear function. All three were tried – while the results achieved by using the rbf were slightly better, I have used *Linear* kernel function because it is computationally faster with almost similar results.
* One to all vs One to One: Again, both were tested, and better results were achieved using one to all. In one to all, we basically train a binary classifier for each individual class. So, n class means n classifiers.
* Lastly, the BoxConstraint parameter was chosen to be optimized throughout training using the OptimizeHyperparameters call.

For extracting the HOG features cell size 8x8 and 16x16 was tested and 16x16 produced better results as it helps capture more information for a 224x224 image size.

For increasing the overall computational speed and utilizing all the CPU cores available – parallel pooling was implemented. The results will be shared in the next section.

## Deep Learning Approach

In the deep learning approach, many different models were researched to understand the underlying theory. Namely YOLOv8, Inception v3, Resnet18, Resnet34 and Resnet50 were Investigated. I was personally unable to run any of these models on my local machine, even on the 20-class subset. From my understanding, the combination of MacBook’s ARM architecture is not ideal for MATLAB and such complex transfer learning models are infeasible without appropriate infrastructure.

So basically, I wrote a simple CNN and tried different depths of the CNN network in my trial-and-error method. The final model I used for Experiment 2 and 4 is as shown below:



Figure 1: CNN Architecture Used

The explanation for each layer used is as follows [7]:

* Input Layer: This layer takes in the raw image data as input and stores its original shape and pixel values often as matrices.
* 2d Convolutional Layer: Applies a set of filters to the input layer that helps in capturing the local image features like edges, textures, or shapes.
* ReLU Activation Layer: Helps in introducing non-linearity to the model by using the rectified linear unit function. This allows the model to learn more complex patterns.
* Pooling Layer: Reduces the width and height of the input features to reduce the overall computation needed in the network. This also helps in detecting features that are invariant to scale and orientation.
* Fully Connected Layer: Neurons in this layer are fully connected to all the activations in the previous layer and are thus used for classifying the image based on the extracted features.
* Dropout Layer: This layer randomly sets a fraction of the input units to 0 at each training iteration. This is done to avoid overfitting.
* Batch Normalization Layer: This layer standardizes/Normalizes the inputs for each mini-batch.
* Softmax Layer: This is applied to the output layer for multi-class classification as it helps in converting the final features into probabilities for each class.

Together these layers help the CNN network to extract features, perform non-linear transformations, and then do the final classification.

In summary a CNN taken an image input then feeds its pixel information through multiple layers. In the process, capturing patters from simple edges to complex boundaries. CNNs can learn these features from raw image data thus are very effective in handling image classification and object detection tasks.

There are many parameter options available for training a CNN and I tried multiple different combinations to land on the final ones used below:

A screenshot of a computer code

Description automatically generated

Figure 2: CNN parameters used

A piecewise learning rate method was used to drop the learning rate by 0.5 using drop period as 10. Maximum 30 epochs with batch size 64 and validation patience 3 was used to reduce the overall time taken for training as well as reduce the chances of overfitting.

Many other combinations of hyperparameters were tried which are not summarized here but are included and commented out in the submitted code files attached to the submission.

# Presentation of Resulsts

## Experiment 1 – Classic ML Classifier

The results achieved by using the SVM classifier with Linear kernel function and HOG feature extraction with cell size 16 and image size 224x224 is below:

A screenshot of a document

Description automatically generated

Figure 3:Training Results SVM – full image

The most accurate result - 3% overall accuracy – for the full image size was achieved using the Linear kernel function in SVM along with onevsall parameter. Below is the graph of the objective function optimization over epochs.

A graph of a function

Description automatically generated

Figure 4: Objective function for SVM - BoxConstraint

Learning: The SVM model architecture is easily understood. Using onevsall for a multi-class classification problem makes it very interpretable. Although, low accuracy achieved was also because of the low complexity of handling image data. The model is unable to generalize well on an extensive dataset.

## Experiment 3 – Bounding Box (SVM)

The results achieved using the same model from Experiment 1 but this time using the bounding box information only and not the entire image is below:

A white paper with black text

Description automatically generated

Figure 5: Training results SVM - bounding box

The overall accuracy achieved was 11.19%, which is a significant increase from 3% in the previous experiment. The only thing changed was using only the bounding box image data than using the full image.

Below is the BoxConstraint optimization graph produced during training:

A graph of a function

Description automatically generated

Figure 6: Objective function for SVM - BoxConstraint

Learning: This experiment helped learn that using bounding box information with SVM increases prediction accuracy because now the classifier can focus on more relevant information. We remove the noise and enhance the features that the model should focus on and learn.

In the process, we also simplify the problem space by reducing the variability in data per class. Also, much faster training and convergence because less more important data was fed to the model.

## Experiment 2 – Deep Neural Network

For this experiment first a transfer learning approach was tested which was not executed in the 10-hour time available on Nutanix. Then a simple CNN was tested that achieved a low accuracy. Then a more complex CNN with increased depth helped achieve a relatively higher accuracy.

The model used, along with all parameters is mentioned in the Methodology section. It is important to note that the results achieved in Experiment 2,4, and 5 include steps for *data augmentation*. I used X and Y translation, random rotation, scaling and reflection for training the CNN models. Without data augmentation the accuracies achieved were significantly lower. Thus, the importance of data augmentation techniques for each specific class was researched and understood in the development process of these experiments.

Results below:

A graph with a number of points

Description automatically generated with medium confidence

Figure 7: CNN full image results

The confusion matrix is submitted as a separate file. An accuracy of 6.6% was achieved with the full image information using 4 conv2d layers and 2 dropout layers, 0.25 and 0.35 respectively. Image size of 224x224 with a piecewise dropping learning rate was used.

Learning: A simple CNN can learn the features and classify better than a classic SVM. There are not a lot of options available for SVM models to increase model complexity to achieve more accurate results. But CNN architecture can be restructured based on the problem statement at hand.

Furthermore, CNNs can extract and learn features better than a Decision Trees, KNNs and SVMs.

## Experiment 4 – Bounding Box (CNN)

For experiment 4, the same parameters and model architecture was used for training. The overall accuracy achieved by the model is 20.7%.

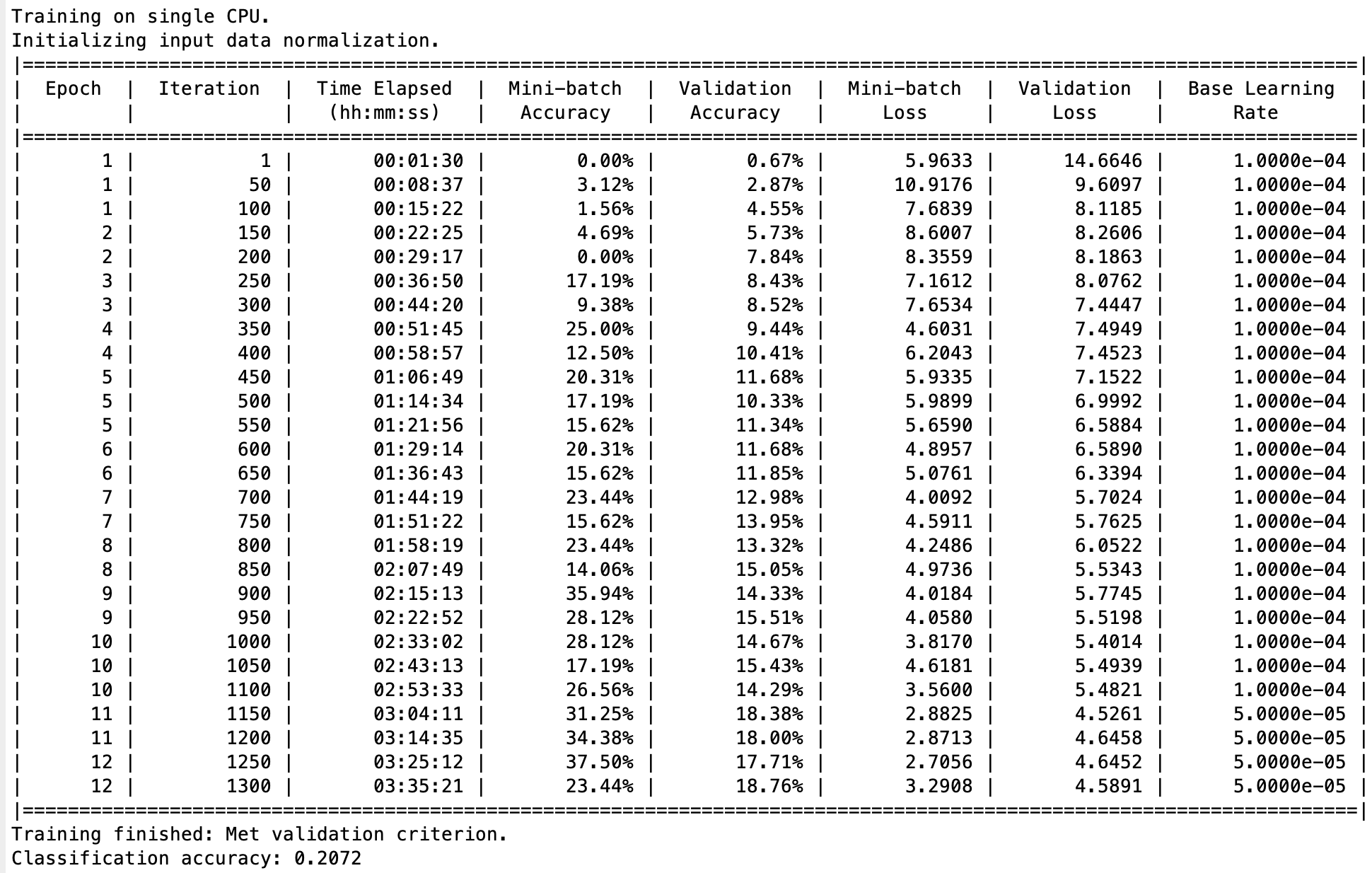


Figure 8: CNN results

Learning: The higher accuracy achieved using just the bounding box information follows the same principle explained earlier. The model can focus on specific relevant information and remove the noise and clutter in the training. We reduce data variability per class and can generalize better.

## Experiment 5

Lastly, the 5-fold cross validation was performed for experiment 4. In this experiment we divide the entire dataset into 5 equal parts and then use different combinations of these parts for training, testing and validation (3:1:1). Because of the computation time required for doing this experiment the maximum number of epochs were lowered to 10 and two different batch sizes were tested, 20 and 64.

Results achieved by the 20-batch size was significantly lower (7%) than the results achieved by batch size 64 (15%). This was unusual to see but could be because of the increased noise in the gradient estimates. Results attached below:

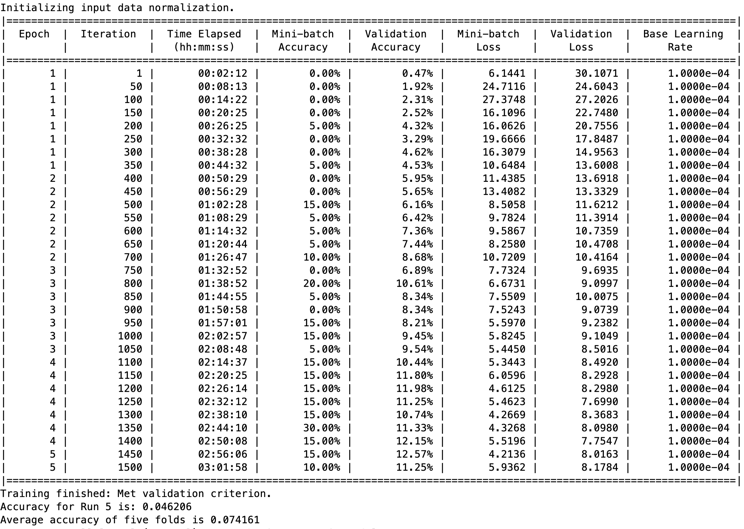


Figure 9: Batch size 20

A screenshot of a computer screen

Description automatically generated

Figure 10: Batch size 64

Learning: Because smaller proportions of the entire dataset were used in different parts, this technique of cross validation helps in understanding and building a more realistic and generalized understanding of the model performance. This technique of cross validation helps in the reduction of bias that may have been produced in the training process. For example, the accuracy achieved in Experiment 4 but during the cross-validation process, just reducing the number of epochs reduced the accuracy by 5 percent but had consistent results throughout the 5 runs. That tells us that no matter which part of the dataset is looked at, it always has a performance of 15%.

Furthermore, by using the validation process in conjunction to the parameter tuning process we can determine the best possible parameters needed for a more realistic, real-world applicable model with higher accuracies.

In summary this process helps us get more reliable, comprehensive, and robust model that does not overfit and is able to generalize well on diverse datasets.

## Learnings and Methods for Future Improvements

In have included below a table summarizing the performance for all experiments.

TABLE 1 Summary of performance

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Acc. | Full Image | |  | | Bounding Box | | | |
| Avg accuracy | | Avg. accuracy | | | |
| EX1 | ***3.09%*** | | EX3 | | ***11.2%*** | | | |
| EX2 | ***6.6%*** | | EX4 | | ***20.7%*** | | | |
| EX5 | Run1 | Run2 | | Run3 | | Run4 | Run5 | Avg. |
| ***15.1%*** | ***15.8%*** | | ***16.9%*** | | ***13.7%*** | ***16.2%*** | ***15.6%*** |

In addition to the learnings mentioned earlier, I have learned:

* Using multiple feature extraction methods in cohesion as a hybrid approach. – Tested it on 20 class subset with not so good results.
* An in-depth understanding of the deep neural network architecture.
* Parameter tuning and cross validation techniques.
* Importance of data augmentation and how a class imbalance can introduce unwanted bias in training.

Future work: The most important next step is to fix the class imbalance using techniques like SMOTE [8] that refers to synthetic minority oversampling technique. This will help remove bias and increase the overall accuracy of the model.

Another important experiment is to run a more complex neural network using transfer learning. A good model would be Renset34 or Alexnet with personalized weights to handle our dataset. The more complex architecture of these models is known to capture more complex features like that of a bird (beak, face, color of a breed, eyes, etc.) with higher accuracy. Because these models use multiple layers are deep, they are computationally expensive but are known to learn and generalize complex images well.

Lastly, I would want to use the trained model for real-time object tracking and test its performance. This is an important step for real-world acceptability.

# Acknowledgements

I would also like to take this opportunity to acknowledge the support and guidance provided by Mr. Roland Goecke, Dr. Min/Maggie Wang and my lab instructor Sahar Pordeli Behrouz. It would have been impossible to achieve any results, without their continuous feedback this semester. I deeply appreciate it.

##### References

1. M. Angaitkar, M. Vrushali Dudhane, M. Raut, M. Siddhesh Gawarshettiwar, A. Professor, and A. Kazi, “Issue 4 volume 4 Title no 5 INTERNATIONAL JOURNAL FOR ENGINEERING APPLICATIONS AND TECHNOLOGY Birds Species Identification from an Image using Machine Learning,” 2021. Accessed: May 05, 2024. [Online]. Available: https://www.ijfeat.org/papers/may20205.pdf
2. M. M. Saeed, “Muhammad-MujtabaSaeed/Caltech-Birds-Classification,” *GitHub*, Feb. 01, 2024. https://github.com/Muhammad-MujtabaSaeed/Caltech-Birds-Classification?tab=readme-ov-file (accessed May 05, 2024).
3. Sivaramakrishnan, “SK124/Bird-Classification-Using-CNNs,” *GitHub*, Apr. 24, 2024. https://github.com/SK124/Bird-Classification-Using-CNNs (accessed May 05, 2024).
4. T. Chakraborti, B. Mccane, S. Mills, and U. Pal, “Collaborative Representation based Fine-grained Species Recognition.” Accessed: May 05, 2024. [Online]. Available: https://tapabrata-chakraborti.github.io/docs/papers/ivcnz.pdf
5. K. Mittal, “A Gentle Introduction Into The Histogram Of Oriented Gradients,” *Medium*, Dec. 21, 2020. https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa
6. “An Idiot’s guide to Support vector machines (SVMs) R. Berwick, Village Idiot SVMs: A New Generation of Learning Algorithms.” Available: https://web.mit.edu/6.034/wwwbob/svm.pdf
7. GeeksforGeeks, “Introduction to Convolution Neural Network,” *GeeksforGeeks*, Aug. 21, 2017. https://www.geeksforgeeks.org/introduction-convolution-neural-network/
8. C. Maklin, “Synthetic Minority Over-sampling TEchnique (SMOTE),” *Medium*, May 14, 2022. https://medium.com/@corymaklin/synthetic-minority-over-sampling-technique-smote-7d419696b88c#:~:text=Synthetic%20Minority%20Over%2Dsampling%20TEchnique%2C%20or%20SMOTE%20for%20short%2C (accessed May 05, 2024).