Used Car Image Classification

Sean Davis

The University of Texas at Austin

November 10, 2024

Abstract

Modern day applications that can classify objects within an image are improving in accuracy and becoming more accessible as time goes on. This project explores how well a model based on convolutional neural networks (CNNs) trained on images of different car models obtained from online car listing platforms can perform when classifying the vehicles given an image. Techniques such as Global average pooling, batch normalization layers, dropout layers, and different optimizers are explored to see how they affect the efficiency and performance of the model. This project could serve as a basic guide to different methods and techniques related to CNNs and how they affect the model as a whole.

1 Introduction

Computer vision is a critical piece of autonomous vehicles. There is much contemporary research on the ability of vehicles to detect obstacles, signs, and generally perceive their surroundings(Tang et al., 2023). These vehicles need to have a computer vision model that is able to accurately discern a pedestrian from a truck or a lamp pole(Grigorescu et al., 2020). In a similar fashion, there is a lot of headway when it comes to image classification of different objects. There are applications that can identify and classify objects just given an image, such as Google Lens. These applications can identify a wide variety of objects, such as plants, animals, star formations, and much more. Inspired by the previously mentioned technology, this project aims to train a computer vision model to be able to classify cars by model, given an image of a car. Similar to the previously mentioned applications, a model such as the one proposed would allow users to identify cars with just a picture or image, whether it is taken in real life, in a movie, or in any other scenario.

The goal of this project is to see how effective a CNN will be when trained and tested against our database of images, and what attributes can optimize its performance.

2 Research Background

Object detection and classification are probably the main focus of computer vision models within autonomous vehicles, which allows them to distinguish between obstacles in the environment. For instance, Chen et al. (2015) demonstrated how CNNs could be applied to traffic sign recognition, achieving high accuracy in classifying various traffic signs under various different conditions concerning image quality and attributes.

Beyond autonomous driving, image classification tasks have improved iteratively over time. Previously mentioned is Google Lens, which uses CNNs to classify a variety of objects in images. Aytar et al. (2016) explored how deep learning models for generalized image recognition tasks and CNNs could be trained on large datasets to provide real-time classification for users. These advancements have led to more robust and versatile image recognition systems, which, similar to Google Lens, can identify objects across a wide variety of categories.

When it comes to the classification of vehicles, He et al. (2019), applied a convolutional neural network architecture for car model classification, among a variety of other objects, and demonstrated the effectiveness of CNNs in distinguishing between similar vehicle models. Their approach was based on training a network using a large dataset of images, which is comparable to the methodology used in this project.

3 Data

The dataset used for this project is made up of images of cars posted on used car websites such as cars.com and Autotrader. I was able to scrape these sites for images of cars based on their model. 9 different models of cars were obtained, including: Cadillac Escalade, GMC Sierra, Honda Civic, Jeep Wrangler, Porsche Cayenne, Ram 1500, Tesla Series 3, Toyota Camry, and Toyota Rav4. Models ranging from the years 2020 to 2024 were selected. Images of cars on used car websites are handy for car detection in that they show off as much of the vehicle as possible. In order to show the condition of the vehicle, images must be posted of the front, back, side, and other angles of the vehicle. Between images, there are also various changes in lighting, distance, and background, making it very unlikely that two images of the front of a Toyota Camry are going to be the exact same, which can enhance model robustness and reduce overfitting (Shorten and Khoshgoftaar, 2019 and Wang et al., 2022)).

3.1 Preprocessing

I was able to obtain around 300 images for each model of car. Ideally, a larger dataset would be better, but I thought this would work for the purposes of this project. The dataset was then split into training and validation sets with an 80/20 split. This will help prevent our model from overfitting according to all the images, and make the model overall more robust(Gholamy et al, 2018).

Before we train the model with images, the images have to be rescaled. This means that pixels within the images must be normalized between a value of 0 and 1(Lecun et al., 1998). This is a crucial step to take in order to improve the effectiveness and efficiency of the model. There are many extra steps that you can take when it comes to preprocessing images, and a few will be covered in the next section.

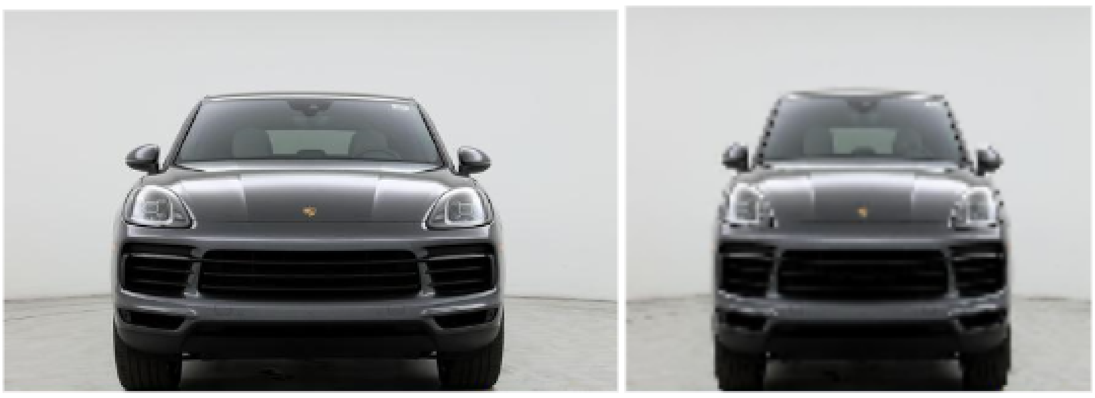

Figure 1: Normal image on the left, image after rescaling to right.

4 Method Backgrounds

4.1 CNN Architecture

The base CNN that is used to start this project has 2 convolution layers, 2 pooling layers, a flattening layer, and 3 fully connected layers. The architecture is similar to the image in the figure below(Mishra, 2024).

A convolutional layer consists of a set of kernels that convolve over the input image to produce an output feature map. The kernels slide over the input image and computes the dot product between the kernel and the corresponding portions of the image, resulting in a scalar value(Alzubaidi,

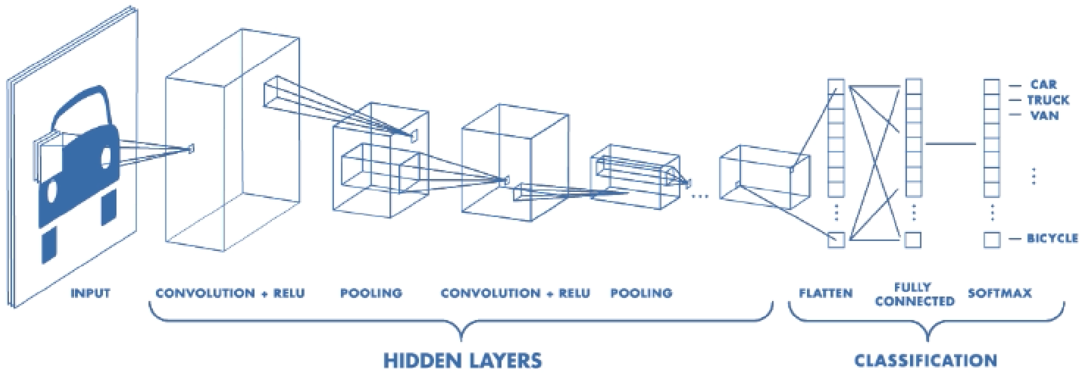

Figure 2: Basic CNN architecture

et al. 2021). ReLU activation is performed with the convolutional layers. ReLU introduces non-linearity within hidden layers of the neural network(Gholamalinezhad, 2024). Without non-linearity, the network would be equivalent to a single-layer model, no matter how many layers it has, as the output of each layer would just be a linear combination of the inputs. Different choices of activation functions within a model can significantly impact model performance, however, ReLU usually outperforms alternatives like sigmoid and tanh in terms of both accuracy and training efficiency(Zhang et al., 2023).

The pooling layer reduces the spatial dimensions of feature maps, helping to decrease computational cost and control overfitting by extracting valuable features while discarding details not seen as important(Yamashita et al., 2018). The feature maps from the final pooling layer are flattened into a 1D array and passed through fully connected layers, where they are mapped to the final output, which in this case is the 9 models of cars. There are several different activation functions commonly used to output final predictions on observations. Softmax is used in cases where there are more than 2 output classes, so it is used in these models(Agarap, 2017).

For evaluation metrics, we measure the accuracy and loss of the model on the training and validation datasets. ADAM is used as the optimizer during training, which adapts the learning rate during training to increase the probability of convergence and improve stability(Dogo, 2022). We also use early stopping during model training. This means that if the model trains for more than 5 epochs without accuracy improving or loss decreasing, we stop training the model. This helps to prevent overfitting the model on the training dataset(Olamendy, 2023).

4.2 Image Augmentation

As mentioned previously, the images used in training the model were rescaled. Building upon that, there are many image augmentation techniques available to us. To view the effects of these techniques, we created two datasets to train our model on. One dataset contained images that were only rescaled, and the other contained images that underwent extra augmentation techniques. These included image rotation, horizontal flips, and shifting the height and weight of the image.

After training the model on these datasets, they had fairly similar results(Table 1), so perhaps a middle ground might see better performance. The dataset was augmented to include rescaling, slight rotation of the images, and random horizontal flip.

When running the model, this set of images gave us the best performance, and consequently was the dataset used for the rest of the models.

	Training accuracy	Training loss	Validation accuracy	Validation loss
Rescale only	.9744	.1374	.5443	1.8082
Most Augmentation	.7069	.8610	.5785	1.7709
Some Augmentation	.9426	.1878	**.6139**	**1.3792**

Table I: Metrics when comparing data augmentation methods

5 Methods

5.1 Adding Layers and Units

The first model created contained extra convolutional layers and units within layers. Two convolutional layers and pooling layers were added to the existing model and the number of units within each convolutional layer was increased from 32 to 64. Likewise, the number of units within the fully connected layers were doubled, from 32 and 64 to 64 and 128. I was curious as to whether this would allow the model to capture more information from the image, different units contain information that varies in importance when it comes to accurately classifying images(Bau et al., 2020).

5.2 Batch Normalization

The second model contained batch normalization layers between the convolutional layers and fully connected layers. Batch normalization addresses the issue of internal covariate shift by normalizing the inputs to each layer so that they have a consistent distribution. This allows for higher learning rates, better initialization, and improved training efficiency(Ioffe, 2015). By reducing dependencies on specific parameter scales, it stabilizes the training process, making it easier to train deeper networks that would otherwise struggle with gradient-related issues.

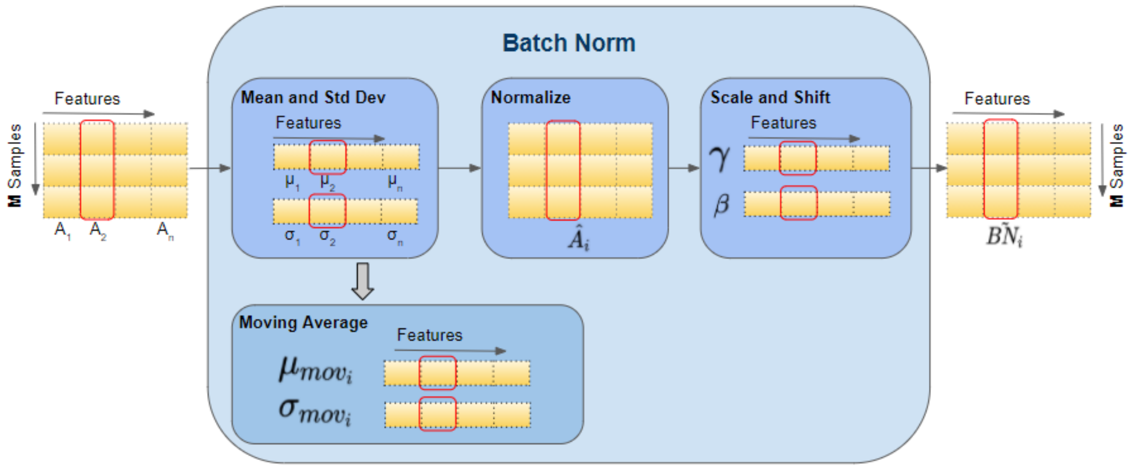

Figure 3: Workflow of a batch normalization layer(Doshi, 2021).

5.3 Dropouts

In this model we introduced dropout layers in between the fully connected layers. Adding dropout layers is a regularization technique aimed at reducing the chance of the model overfitting by randomly dropping out or disabling a fraction of the neurons in the network along with their connections. This forces the network to avoid relying too heavily on specific neurons and encourages it to develop a more robust set of features across different neurons(Srivastava et al. 2014). This model contained 2 dropout layers, with each dropping 10 percent of the neurons, leaving 80 percent of the neurons remaining when the model is finished training.

5.4 Global Average Pooling

Instead of using fully-connected layers, a model using global average pooling compresses each feature map into a value, using the average of values within the feature map(Olu-Ipinlaye, 2022).

A large advantage of global average pooling is that it reduces the total parameter count within the model. For example, previous models have had parameter counts in the millions, while the model using global average pooling only contains around 55,000 parameters. A lower number of parameters can help the model prevent overfitting and can be more efficient. Additionally, global average pooling offers enhanced spatial invariance, as it focuses on the general presence of features rather than their exact locations within the feature map. This approach can lead to improved generalization in real-world applications, where exact spatial locations may vary. However, it has been found that while global average pooling can reduce model complexity, its effectiveness was highly dependent on the diversity of viewing angles in the training dataset, which is a large part of our dataset(Park and Wilson, 2024).

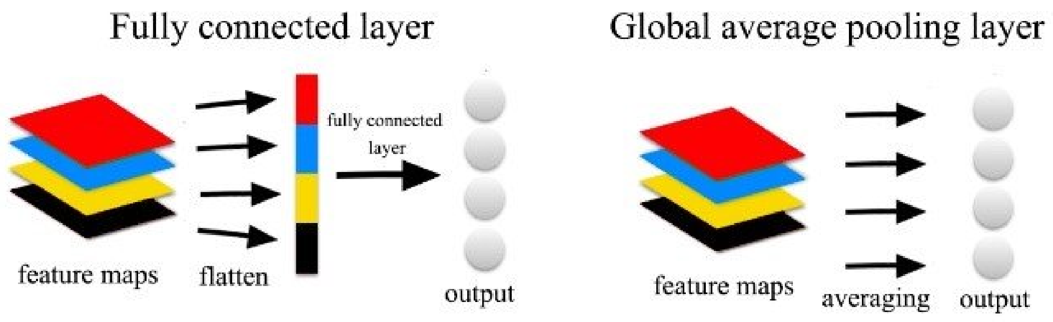

Figure 4: The difference between a fully connected layer and a global average pooling layer(Guo, 2020).

5.5 Duplicating Data

For one model we duplicated our database to train the model on, meaning our dataset now had a pair of each image. While this sounds like it might cause the data to overfit, as we're training the model on similar images, the data augmentation steps we made previously made it extremely unlikely that the model would receive the same image multiple times, due to the random rotation and horizontal flips the dataset was edited with. This is done in order to train the model with more data than what it was being trained with previously. Increasing the amount of data available to train a model with, especially diverse data, is essential to getting good performance from a model. This is especially true for deep learning models such as CNNs(Roh, 2019).

5.6 Overfitting

You may have noticed that almost all of the methods and techniques introduced in this paper are to serve a specific purpose: to prevent overfitting. The term overfitting describes a case when a model is trained on the same data or images over time, and the model memorizes the input, instead of generalizing features within the input(Xiao et al., 2021). A model that has been overfit on the training data will produce good training accuracy and loss metrics, but it will perform poorly when tested on a validation dataset, or any dataset that the model has not seen. This is due to the problem of memorizing features instead of generalizing features. This is a common occurrence with CNNs and all different types of deep learning models.

The charts below(figure 5) give an example of a model that has been overfit and one that has stopped training before it has been overfit. The overfitted model achieves perfect training accuracy and loss within 10 epochs. However, its validation accuracy plateaus at around the same mark, and it's around there where the validation loss stops decreasing and starts to increase. This is due to the model memorizing the features from the training set over and over instead of generalizing the features.

The other model is stopped early to prevent overfitting, and these graphs are more desirable when it comes to performance. The training accuracy and loss continually increases/decreases, with the validation accuracy and loss following suit behind the training metrics. Ideally, the model can be trained with enough epochs to where the training and validation accuracy and loss eventually converge to a value.

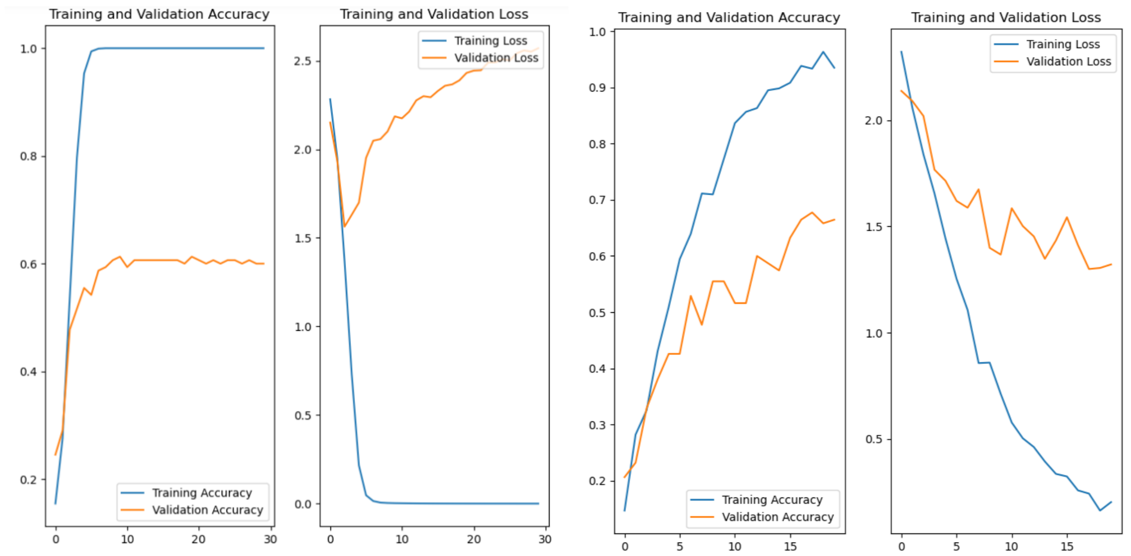

Figure 5: Model that has been overfit vs model that stopped training before overfitting could occur.

6 Results

After training the model on the training datasets, we can view the results of testing them on the validation datasets, using the evaluation metrics in table 2.

	Training acc	Training loss	Validation acc	Validation loss
Base model	.9426	.1878	.6139	1.3792
Base model + Extra layers	.9526	.1313	.6524	1.3169
Base model + Batch norm	**.9854**	.0706	.1763	5.6572
Base model + Dropout layers	.9470	.1663	.7615	1.2559
Base model + Global avg. pool	.3127	1.8722	.2194	2.0503
Base model + Extra train data	.9741	**.0697**	.6505	1.5934
Best model	.9757	.1234	**.7903**	**.9344**

Table II: Evaluation metrics from each version of the model.

When adding extra layers and units to the model, the model performed slightly better than the base model, gaining 4 percent validation accuracy. While the model was able to achieve a higher accuracy, it is important to note that a model that receives extra information as input has a greater risk of overfitting. So it is important that early stopping is enforced within the training of each model. The model took around twice as long to train, which makes sense, with the amount of units within each layer doubling in size.

The model with batch normalization layers achieved the best training accuracy of all the models, and a very low training loss. However, it performed the worst of all models when tested on the validation dataset. This looks like a classic case of a model overfitting the training dataset. The most likely cause of this is the relatively small dataset the model is training on. Since batch normalization calculates statistics based on mini-batches, an already small dataset may cause statistics that are skewed or have a lot of noise(Santurkar, 2019). Using batch normalization also increased the time it took to train the model. With the base model, each epoch took around 10 seconds to train. While the model that included batch normalization needed over 30 seconds for each epoch, which is a large increase for such a drop in performance. It seems that batch normalization isn't a technique that fits this kind of classification problem, or it at least needs to be trained on much more data.

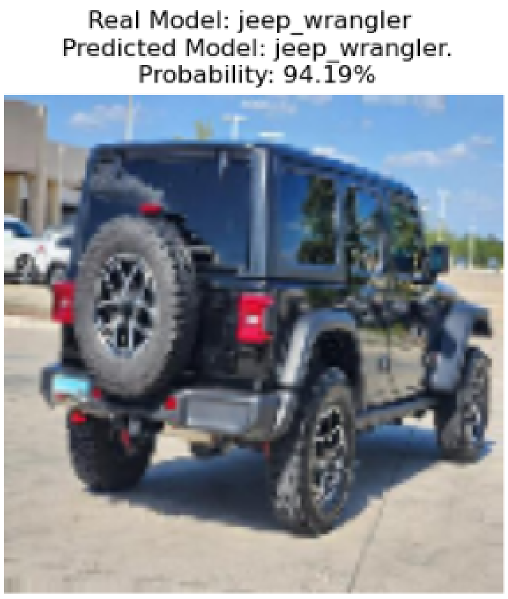

Figure 6: Example output from the final model.

Similarly, the model using global average pooling performed quite poorly. This is probably due to the enhanced spatial invariance, meaning that the locations of features within the feature do matter in the grand scheme of things, and the presence of the feature by itself isn't enough to make the model perform well enough. On a more positive note, this model was the fastest model to train by far, only taking half as much time to train as the base model.

The model that introduced dropout layers performed the best of all models by far. This probably occurred due to its resistance against a model overfitting, which is probably helpful considering the relatively small dataset we had to train the model with. Also, dropping neurons that contain a variable amount of information forces the model to generalize features found within the training data, which will make the model more robust when it is put to the test against the validation dataset.

The model that received extra training data performed slightly better than the base model, but not as much as I would have hoped. While the data augmentation performed during the preprocessing step would bring some variance within the images that make up the dataset, of course, it would be more beneficial to find new images to train the model on, as it would allow more features to be found by the model and force the model to generalize more.

Finally, the final model included extra layers and units, dropout layers, and it was trained on the extra training data. This model was able to achieve the best performance of any model with any combinations of layers or techniques edited within the model. The model was able to achieve a validation accuracy of 79 percent and a loss of under 1, which is far improved from the base model.

6.1 Optimization

After determining the model architecture that best suits this use case with our image dataset, I wanted to try different optimization methods within the model other than ADAM, to see how it would perform. The other optimization methods chosen were AdamW, Adamax, RMSprop, and SGD.

SGD is a very basic optimizer used for a variety of machine learning problems. SGD has a fixed learning rate and it can be difficult to escape local minima(Tian, 2023). AdamW decouples the weight decay from the learning update, which could lead to better generalization and be faster than Adam(Loshchilov, 2019). RMSprop uses an adaptive learning rate, which allows modifications to the learning rate during training(Ruder, 2017). Adamax uses an infinite norm in updates instead of the L2 norm(Kingma, 2017).

	Training acc	Training loss	Validation acc	Validation loss
Adam	.9757	.1234	**.7903**	**.9344**
SGD	.8141	.5714	.6161	1.2988
Adamax	.9643	.2310	.7000	1.0394
RMSprop	.9822	.0823	.7194	1.6459
AdamW	**.9846**	**.0487**	.7129	1.7745

Table III: Evaluation metrics from each optimizer used within the model.

Unfortunately, none of the other optimizers performed even close to as well as Adam. This could be that we fine-tuned the model in accordance with using Adam, however, Adam is a very standard optimizer to use for machine learning and deep learning problems and works well across a wide variety of tasks. SGD performed quite poorly compared to the rest, which could be due to its static learning rate.

7 Conclusion

The models built and trained within this project can be used to classify car models given images of cars with variable backgrounds, angles, and lighting. After trying out methods and techniques related to deep learning and CNNs such as different model architectures and optimizers, the final model achieved a much higher accuracy on our test dataset than the original model. Introducing dropout layers to the models ended up being the modification that most improved the performance of models.

In terms of future research, the most pressing issue would be to increase the size and diversity of the image dataset. Being able to train the model on more data including more models of vehicles, the model would probably perform better, and there would be a wider variety of vehicles the model would be able to identify and classify.

8 References

8.1 Scholarly references

Agarap. (n.d.). Deep learning using rectified linear units (ReLU). https://arxiv.org/pdf/1803.08375

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., Farhan, L. (2021). Review of Deep Learning: Concepts, CNN Architectures, challenges, applications, Future Directions. Journal of Big Data, 8(1). https://doi.org/10.1186/s40537-021-00444-8

Aytar, Y., Chua, T.-S., Zisserman, A. (2016). Cross-Modal Scene Retrieval: Learning to Compare Image and Text. Proceedings of the European Conference on Computer Vision (ECCV), 271–287.

Bau, D., Zhu, J.-Y., Strobelt, H., Lapedriza, A., Zhou, B., Torralba, A. (2020). Understanding the role of individual units in a deep neural network. Proceedings of the National Academy of Sciences, 117(48), 30071–30078. https://doi.org/10.1073/pnas.1907375117

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L. (2015). Semantic image segmentation with deep convolutional nets and fully connected CRFs. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 3578–3586.

Dogo, E. M., Afolabi, O. J., Twala, B. (2022). On the relative impact of optimizers on convolutional neural networks with varying depth and width for image classification. Applied Sciences, 12(23), 11976. https://doi.org/10.3390/app122311976

Gholamalinezhad, H., Khosravi, H. (n.d.). Pooling methods in Deep Neural Networks, a review. https://arxiv.org/pdf/2009.07485

Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. Journal of Field Robotics, 37(3), 362-386.

Guo, Y., Xia, Y., Wang, J., Yu, H., Chen, R.-C. (2020). Real-time facial affective computing on mobile devices. Sensors, 20(3), 870. https://doi.org/10.3390/s20030870

He, K., Zhang, X., Ren, S., Sun, J. (2019). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

Ioffe, S., Szegedy, C. (2015, March 2). Batch normalization: Accelerating deep network training ... https://arxiv.org/pdf/1502.03167

Kingma, D. P., Lei Ba, J. (2017, January 30). Adam:Amethod for stochastic optimization. https://arxiv.org/pdf/1412.6980

Loshchilov, I., Hutter, F. (2019, January 4). Decoupled weight decay regularization. https://arxiv.org/pdf/1711.05101

Park, S., Wilson, J. (2024). "Analysis of pooling techniques in deep learning models for automotive classification tasks." Neural Networks, 159, 328-341.

Roh, Y., Heo, G., Whang, S. E. (2019, August 12). A survey on data collection for Machine Learning. https://arxiv.org/pdf/1811.03402

Ruder, S. (2017, June 15). An overview of gradient descent optimization algorithms. https://arxiv.org/pdf/1609.04747

Santurkar, S., Tsipras, D., Ilyas, A., Madry, A. (2019, April 15). How does batch normalization help optimization? - arxiv. https://arxiv.org/pdf/1805.11604

Shorten, C., Khoshgoftaar, T. M. (2019). A survey on image data augmentation for Deep Learning. Journal of Big Data, 6(1). https://doi.org/10.1186/s40537-019-0197-0

Srivastava, N., Hinton , G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from ... https://www.cs.toronto.edu/ rsalakhu/papers/srivastava14a.pdf

Tang, Y., Zhao, C., Wang, J., Zhang, C., Sun, Q., Zheng, W. X., Du, W., Qian, F., Kurths, J. (2023). Perception and navigation in autonomous systems in the era of Learning: A survey. IEEE Transactions on Neural Networks and Learning Systems, 34(12), 9604–9624.
https://doi.org/10.1109/tnnls.2022.3167688

Tian, Y., Zhang, Y., Zhang, H. (2023). Recent advances in stochastic gradient descent in deep learning. Mathematics, 11(3), 682. https://doi.org/10.3390/math11030682

Wang, H., Liu, Y., Zhang, W. (2022). "Optimal data augmentation strategies for vehicle classification with limited training data." Pattern Recognition Letters, 155, 121-130. Why 70/30

or 80/20 relation between training and testing ... (n.d.).
https://scholarworks.utep.edu/cgi/viewcontent.cgi?article=2202context=cs$_techrep$

Xiao, M., Wu, Y., Zuo, G., Fan, S., Yu, H., Shaikh, Z. A., Wen, Z. (2021). Addressing overfitting problem in deep learning-based solutions for next generation data-driven networks. Wireless Communications and Mobile Computing, 2021(1). https://doi.org/10.1155/2021/8493795

Yamashita, R., Nishio, M., Do, R. K., Togashi, K. (2018). Convolutional Neural Networks: An overview and application in Radiology. Insights into Imaging, 9(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9 Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to

document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791. keywords: Neural networks;Pattern recognition;Machine learning;Optical character recognition software;Character recognition;Feature extraction;Multi-layer neural network;Optical computing;Hidden Markov models;Principal component analysis, Zhang, K., Li, M.,

Yang, R. (2023). "Comparative analysis of activation functions in CNN-based vehicle classification systems." Neural Computing and Applications, 35(2), 1123-1138.

8.2 Non-scholarly references

Mishra, M. (2020, September 2). Convolutional Neural Networks, explained. Medium.
https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

Olamendy, J. C. (2023, December 8). Understanding early stopping: A key to preventing overfitting in machine learning. Medium. https://medium.com/@juanc.olamendy/understanding-early-stopping-a-key-to-preventing-overfitting-in-machine-learning-17554fc321ff

Olu-Ipinlaye, O. (2022, September 30). Global pooling in Convolutional Neural Networks. Paperspace by DigitalOcean Blog. https://blog.paperspace.com/global-pooling-in-convolutional-neural-networks/

Doshi, K. (2021, May 29). Batch norm explained visually-how it works, and Why Neural Networks Need it. Medium. https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739