SwimSafe: a Computer Vision Pool Alarm

Luke Hansen
Stanford University
lrhansen@stanford.edu

Abstract

In the United States, children ages 1–4 are more likely to die from drowning than any other cause except birth defects. Many children who drown gain unsupervised access to a pool, and are not expected to be near it. Better surveillance of home pools could save the lives of many children by alerting unaware parents that a child has entered their pool. This project presents a computer vision application which addresses this problem: SwimSafe is a binary classifier which identifies an image as having an occupied pool or an unoccupied pool. SwimSafe could be used monitor home pools the application can ensure that no unsuspecting children enter the pool. In this work, various methods are explored to create SwimSafe, including fine-tuning a pretrained model and using the output of an off-the-shelf image segmentation model to classify images. The best performing model was a pretrained model with a MobileNetv2 backbone. It achieved an accuracy of 0.956 and an F1 score of 0.966 on the test set. Computer vision techniques show promise in helping reduce the number of drownings in unsupervised home pools.

1. Introduction

320,000 people die from drowning worldwide each year [3]. Children are particularly vulnerable: in the United States, children ages 1–4 are more likely to die from drowning than any other cause except birth defects [2]. 87% of these drownings occur in home pools or hot tubs [3]. Furthermore, for every fatal drowning, there are 5-10 non-fatal drowning injuries, often with lifelong consequences. Many children who drown gain unsupervised access to a pool, and are not expected to be near it. Better surveillance of home pools could save the lives of many children by alerting unaware parents that a child has entered their pool. This project presents a computer vision application which addresses this problem: a binary classifier was developed which identifies an image as having an occupied pool or an unoccupied pool.

Two different approaches to creating the binary classifier are described: 1) fine-tuning a pretrained network using training data (called SwimSafe) and 2) implementing an algorithm which uses the output of an off-shelf image segmentation neural network (which identifies pools and swimmers in images) to classify the image (called SegSwimSafe).

Such classifiers could be implemented to monitor a video stream of a pool and to alert pool owners if someone has entered the pool. When pool-owners want to use the pool, they can toggle the application. When they are not intentionally using the pool, the application can ensure that no unsuspecting children enter the pool.

2. Related Work

Deep learning approaches have shown remarkable efficacy in many domains, including natural language processing and computer vision. Unfortunately, these deep learning methods require a significant amount of data. Finding sufficient high-quality data was challenging for this project, so I investigated research that has confronted difficult, similar tasks with small datasets. One group of researchers created an object detection algorithm which identifies the location of swimmers in competitive races [8]. They accomplished this feat using transfer learning, which involves re-using a model trained for a certain task on a new task. They fine-tuned a pretrained YOLO network on a relatively small dataset, achieving impressive results.

Transfer learning for image classification requires a pretrained "backbone" network. As further described in the Methods section, various model architectures (all trained on ImageNet [6]) were experimented with as the backbone of the SwimSafe model, including:

1. VGG-19: A 19 layer network that uses 3x3 convolutions and 2x2 maxpool blocks (∼38 million parameters) [12].

2. ResNet-50: a 50 layer network which leverages residual connections (∼23 million parameters) [7].

3. MobileNetv2: a 53 layer network that uses depthwise-separable convolutions to create a model that can be

run on mobile devices (∼3 million parameters), [11].

4. NasNetMobile: Researchers used a novel search technique to determine optimal architectures for a network that can run on mobile devices; (∼3 million parameters) [20].

For the SegSwimSafe algorithm, an image segmentation model was needed. Image segmentation refers to the task of assigning each pixel in an image to a category. Researchers at MIT introduced the ADE20 dataset, which contains 25,000 images annotated with 150 classes [19]. In addition to the dataset, they provide various encoder and decoder networks trained on it for public use [18]. One of these models was used in this project (details in the Methods section).

2.1. Existing Solutions

Various current methods used to try to prevent child drownings were also researched. One paper recommends the installation of a barrier around home pools to prevent children from falling in; however, they report that such approaches have proven insufficient, showing a "lack of any significant decline in drowning rates to young children over time" [13].

A company called SwimEye has developed an application that detects drowning individuals using cameras installed under the water [4]. This method is effective, but the cameras are expensive, and are difficult to install. Furthermore, the system only detects individuals once they have been submerged for a long time, giving a limited response time to parents and increasing the risk that lifelong injuries will occur. Another strategy is to use pool alarms which sense perturbations in the water surface. However, one study found that these alarms were prone to giving false positives (for example, if the water was disturbed by rain), and sometimes had slow response times (depending on how far the disturbance was from the sensor) [17].

3. Dataset

The data used in this project came from two sources: 1) the Swimm400 dataset (courtesy of [8]), and 2) images gathered from the Internet.

The Swimm400 dataset consists of 403 images taken during various points of swim races, and the corresponding bounding box coordinates of each swimmer in the images. The dataset was also used for this project, but with different labels: each picture was categorized as to whether the pool was empty (when the swimmers were on blocks and in between races) or contained swimmers. Some of the images that contained swimmers were cropped so that no swimmers were visible in the pool. These images were labeled as "empty" to balance the ratio of positive to negative samples.

All these data were taken in an olympic swimming pool, which would likely limit the effectiveness of the SwimSafe algorithm for home pools. To increase the diversity of the training data, I also gathered data from the internet. First, a web-scraper called Bing Image Downloader was used to download images from a Bing search query [1]. However, the overall quality of the images that were found was quite low–many were not even relevant to the search query. Given the limited success of this approach, other data collection strategies were attempted. Images from the internet were manually downloaded and categorized.

In total, the dataset contained 676 images (444 positive (swimmers in pool), and 232 negative (empty pool) samples). The dataset was split into train (60%; 404 photos), val (20%; 134 photos), and test (20%; 137 photos) sets.

3.1. Preprocessing and Augmentation

The ImageFolder class (courtesy of Pytorch [10]) was used to load the images into Python. Since the dataset was relatively small, data augmentation strategies were utilized: random rotations were applied to some images; and some images were flipped horizontally. Each image was cropped to have dimensions of 224x224 pixels. In addition to this, the channels in the photo were normalized.

4. Methods

4.1. Approach 1: SwimSafe

Loss function

The Binary Cross Entropy (BCE) Loss function was used to quantify the model's efficacy.

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p(y_i)) + (1-y_i)\log(1-p(y_i))\right)$$

This loss function was chosen for empirical and theoretical reasons. Many successful binary classifiers have been trained using BCE, and, according to information theory, BCE can represent the difference between a predicted probability distribution and a ground truth distribution.

Model Architectures

I experimented with a variety of different pretrained models as the backbone of the SwimSafe model. The four I chose (listed as 1,2,3,4 in the Related Work section) represented a diverse set of models, coming from different stages in the development of deep learning methods (from 2015 (VGG-19) to 2018 (NasNetMobile), with different architectural innovations (ResNet-50 introduced residual connec-

tions) and different constraints (MobileNetv2 and NasNet-Mobile were created for mobile phones). Each model had an additional linear layer which projected the output of the model to 2 dimensions. Only the weights of this additional layer were updated using gradient descent.

4.2. Approach 2: SegSwimSafe

An off-the-shelf image segmentation model was used in SegSwimSafe. I used a pretrained ResNet-50 network as the encoder and the pretrained ppm_deepsup network as the decoder provided by researchers at MIT [18]. The final model classified each pixel in an image into 150 different categories, including the categories of person, pool, and water (as seen in Figure 1). I developed an algorithm that uses the output of this network to classify whether or not a person is in a pool.

This algorithm relies on the observation that if a person has a lot of contact with water, then they are likely in water. For each person, the number of "person" pixels directly above "water" pixels was calculated. If this ratio is above a certain threshold (a hyperparameter called the sensitivity), then the person was counted as being in the water. Only "person" pixels that were directly above "water" pixels were considered because a person should only be considered in the water if the bottom perimeter of the person is in contact with water. (Consider an image of someone standing in front of a pool–they might be surrounded with water, but they are not in the pool. If only the pixels below the person are considered, this problem is mitigated).

The algorithm has the following pseudocode:

1. Run the Image Segmentation model.
2. Use a scipy function to identify all the distinct humans in the output (through the contagious areas of "person" pixels) [16].
3. For each person, count the number of "water" pixels directly underneath them, and divide that value by the total number of the pixels for that person.
4. If this ratio is greater than the sensitivity hyperparameter, classify the person as in the water.

5. Experiments

Each SwimSafe model with a pretrained backbone was trained for 25 epochs with a learning rate of 0.0001 and a batch size of 4. These hyperparameters were determined after trying many learning rates (.01, .001, .0001) and many batch sizes (4, 8, and 16) on a small amount of the training data to determine which converged the fastest. The .0001 learning rate and batch size of 4 worked well for all models, and the performance of all the models plateaued within 25 epochs.

All models were trained with the Adam optimizer [9]. Adam optimization combines the benefits of the AdaGrad and RMS prop optimizers to perform stochastic gradient descent. Much empirical evidence demonstrates that it is a powerful optimization technique because it can improve training times of models through faster convergence and prevent models from getting stuck in local minimums.

SegSwimSafe only had one hyperparameter to choose: the sensitivity. This value was determined by running the model on a subset of the training data (for computational efficiency) with various candidate values (0.001, 0.01, 025, 0.05, 0.075), and selecting the one with the best performance.

6. Results

Metrics

Accuracy and F1 score were used to quantify the models performance:

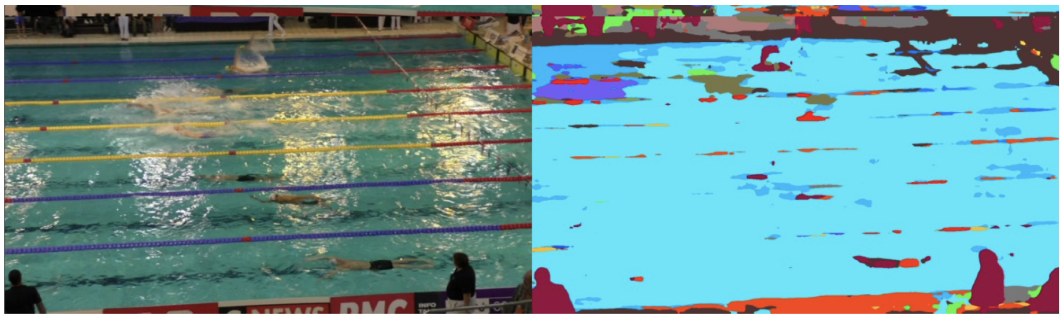
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

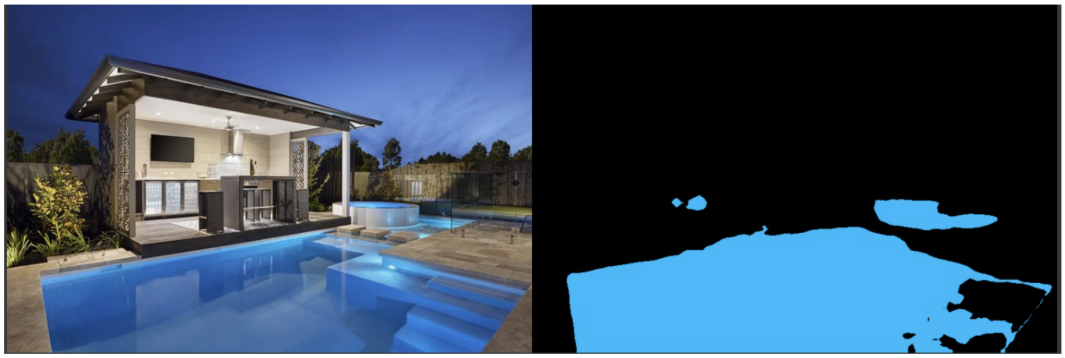
Accuracy provides a rough approximation of model performance that is easy to interpret, but can veil certain weaknesses of a model (especially when there is significant class imbalance). The F1 score helps further gauge a model's performance in a way that is not subject to the distribution of examples. It is the harmonic mean of two competing metrics: precision (correctly predicted positive samples relative to all positive predictions) and recall (correctly predicted positive samples relative to all actual positive samples), and is useful even when the classes are unbalanced (unlike accuracy).

Quantitative Evaluations of SwimSafe Models

Each model's performance can be seen in Table 1. All of the SwimSafe models with a pretrained backbone performed similarly well, with only slight differences in their accuracies and F1 scores. This variation could be due to noise– a bigger test set would be needed to confidently declare one as the best. With that caveat, the model with the ResNet-50 backbone performed the best on the val set, with an F1 score of .983, and an accuracy of .978. Surprisingly, the model with MobileNetv2 backbone performed the best on the test set, with an F1 score of .966 and accuracy of .956. This model had much fewer parameters than ResNet-50 or VGG-19, which would seem to suggest it would have less representational capacity. However, this did not seem to affect the performance of the model. I suspect this can be



(a)



(b)

Figure 1. Output of Off-The-Shelf Segmentation Model (Pixels Colors: Maroon = Person; Dark Blue = Pool; Light Blue = Water) a) an Image an Olympic Pool with All Pixels Classifications Shown. b) An Image of a Home Pool with Only the Pixels Classified as "Pool" Shown

Model	**Val Set**		**Test Set**	
	Accuracy	F1	Accuracy	F1
VGG-19 SwimSafe	0.970	0.978	0.942	0.956
ResNet-50 SwimSafe	0.978	0.983	0.949	0.960
MobileNetv2 SwimSafe	0.955	0.966	0.956	0.966
NasNetMobile SwimSafe	0.955	0.966	0.942	6.956
SegSwimSafe	0.955	0.966	0.912	0.934

Table 1. Comparison of Model performance on Dev and Test Sets

explained by the fact that it was developed in 2018 so the developers could take advantage of deep learning innovations that did not exist when ResNet-50 (2015) and VGG-19 (2014) were developed.

The models designed for mobile deployment also trained much faster (MobileNetv2: 19 sec; NasNetMobile: 34 sec) compared to the other models (VGG-19: 8 m 33 sec; ResNet-50: 5m 15 sec). All models were trained using a Google Colab GPU. The mobile models also had much faster single-image processing speeds, making them superior for a real world-application like video stream monitoring.

Qualitative evaluations of SwimSafe

A saliency map depicts the relevancy of each pixel in an image to the final prediction. I created saliency maps for the VGG-19 model using guided backpropagation, a technique that only backpropagates positive error signals (negative gradients are zeroed out) and only considers positive inputs [14]. The results for some of the predictions can be seen in Figure 2 (in black and white for clarity).

The SwimSafe model with the VGG-19 backbone correctly classified Figure 2 Images a, b, c and d. In Images a and c, the model correctly identified the people in the image as being important to classifying the pool as being occupied. The model also identified features of the pool as being relevant when it was unoccupied as can be seen in Images b and d. Notice that Image b also contained people, but the algorithm did not flag them as being relevant to the classifi-

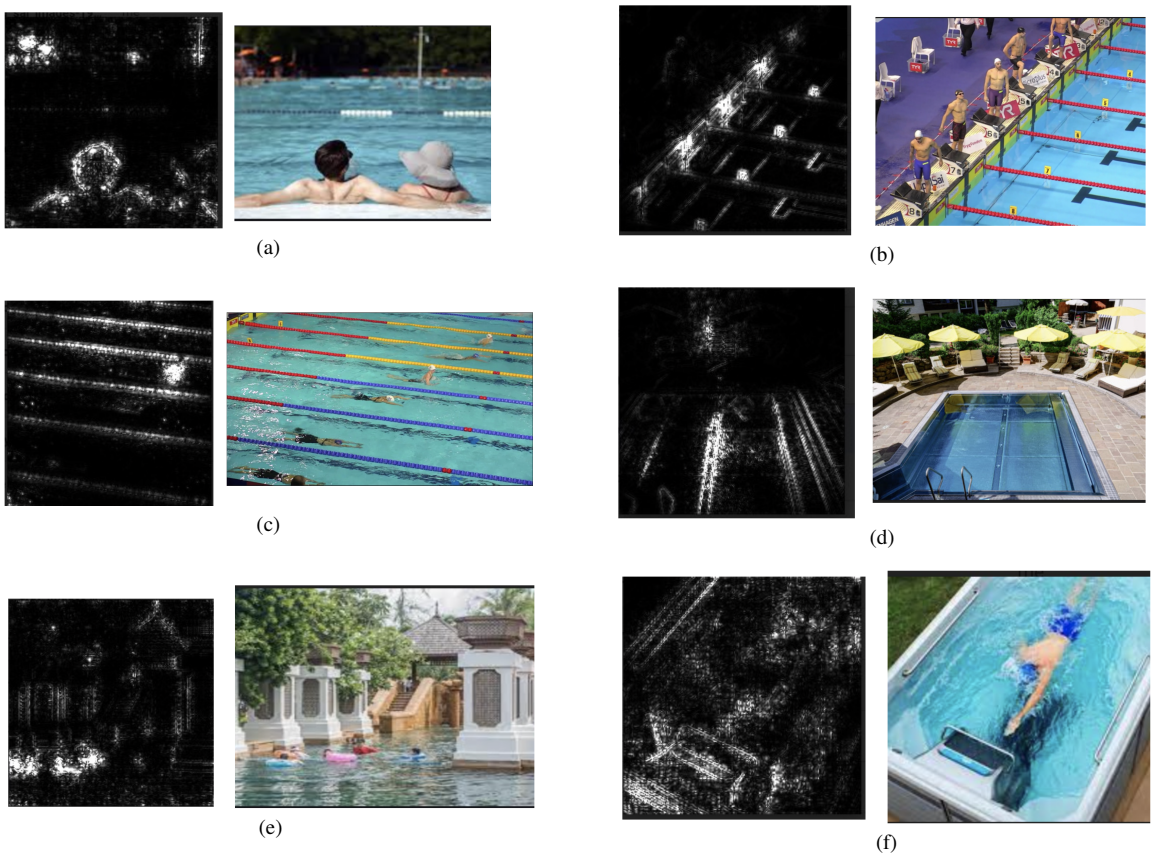


Figure 2. Correct Predictions (Images a, b, c, and d) and Incorrect Predictions (Images e and f) with Saliency Maps

cation. This observation suggests that the model attended to people when they were in the pool (Image a and c), but not when they were outside of the pool (Image b), suggesting the model did not just classify all the images with people as being positive samples–it was not a people detector. I was concerned that the model would learn to classify all images from the swimm400 dataset as being positive samples due to the class imbalance. Techniques used to address the imbalance (cropping out swimmers) appear to be effective as the model did not classify all swimm400 images as positive samples (as can be seen in Image b).

Saliency maps were also created for the incorrect predictions (two of which are Images e and f). In Image e, the model identified the people as being important, but did not recognize that they were in the pool. I suspect this error was due to the abnormal properties of the pool, having a very different color and shape than other samples. The pool also did not have any of the secondary properties that typically are present in pools but are not necessary for a pool to have (line markers, pool ropes, ladders). There is concern that the model was relying too much on these secondary properties for classifications because there was an overrepresentation of such pools in the training data. As can be seen in Images d and c, the line demarcations were very relevant to recognizing the pool. A larger, more diverse dataset would help ameliorate this problem.

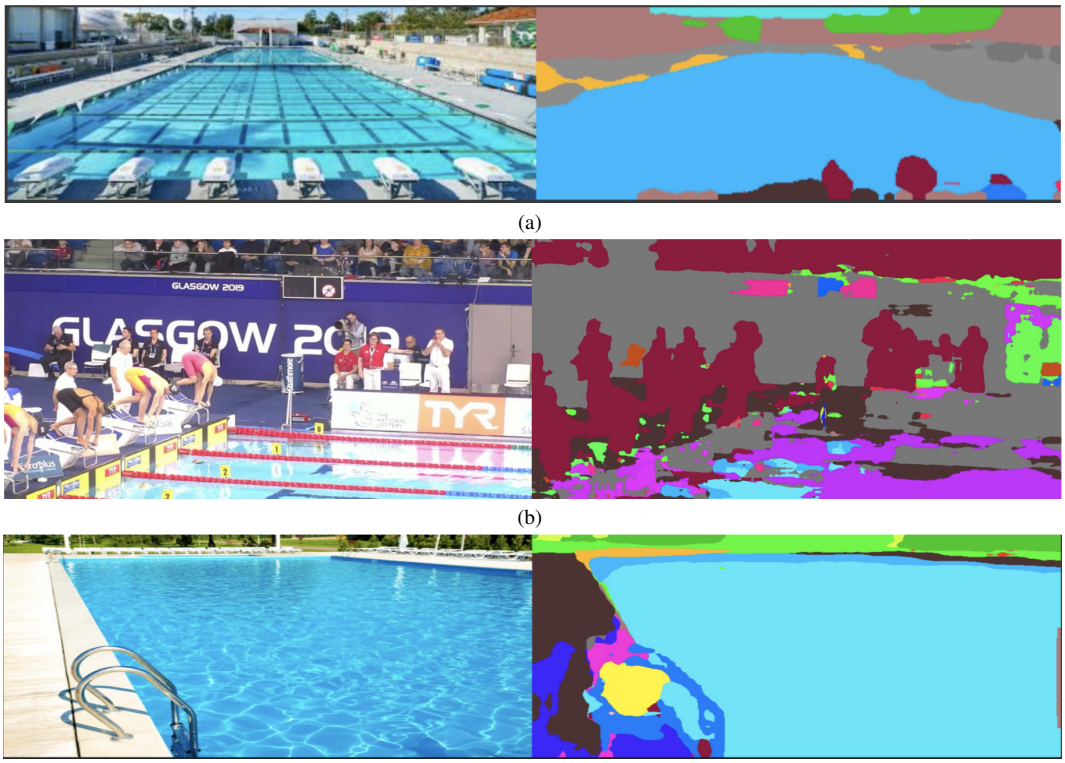
In Image f, the model seemed to have identified the pool as being relevant, but not the person. Though it is hard to know for certain, I speculate that this might have been due to the image having an over-the-head angle of the person in the pool. Viewing people at such an angle was not very well represented in the training set.

Quantitative evaluations of SegSwimSafe

SegSwimSafe performed as well as some of pretrained SwimSafe algorithms on the dev set (accuracy .955; F1: 0.967), but was slightly less effective than the SwimSafe algorithms on the test set having the lowest F1 (0.934) and accuracy (0.912) scores of all of the algorithms.

Qualitative evaluations of SegSwimSafe

The classification mistakes made by the SegSwimSafe were much more scrutable than the mistakes made by the SwimSafe algorithms (given the output of the segmentation



(a)

(b)

(c)

Figure 3. Misclassified Images of SegSwimSafe with Pixel Classifications (Pixels Colors: Maroon = Person; Dark Blue = Pool; Light Blue = Water)

model). Some were due to glaring mistakes by the segmentation model. For example, in Figure 3 Image a, a diving board was identified as human. In another photo (not included), the pool was classified as a wall instead of a swimming pool. Other mistakes were due to more reasonable mistakes made by the segmentation model. In Figure 3 Image b, the segmentation model classified the swimmer's reflection in the water as "person" pixels, causing the image to be misclassified.

The hardcoded algorithm could have been more robust to some of the errors made by the segmentation model. For example, the classification algorithm was very fragile to small, random mistakes made by the segmentation model. In Figure 3 Image c, a very small portion of the image (in the nook of the pool ladder) was categorized as a human. Even though this was a small error for the segmentation algorithm, it had drastic consequences for the classification algorithm because the small area had a high ratio of area to perimeter.

7. Conclusions and Future Work

I present three conclusions based on this research:

1. Consistent with other research, transfer learning can be used to create effective classification models for tasks without significant amounts of training data.

2. End-to-end deep learning models outperformed a hand-crafted algorithm infused with human knowledge. Another confirmation of what Richard Sutton called the "bitter lesson in AI": general methods which leverage computation tend to be more effective than task-specific hand-crafted algorithms [15]. In this work, learning based methods were the most effective, but at the cost of some interpretability of the model output.

3. Computer vision techniques show promise in helping reduce the number of drownings in unsupervised home pools.

This work presents an initial attempt at creating a pool monitoring system that is powered by computer-vision. More, higher quality data (ideally from home pools with cameras already installed) could create a much more effective, robust model that could be deployed with more confidence. More of such data would allow the exploration of more sophisticated computer vision techniques such as Video Action Understanding, instead of framing the problem as a frame-by-frame binary classification task.

8. Acknowledgements

Public codebases used:

1. Saliency Map Generation: https://github.com/utkuozbulak/pytorch-cnn-visualizations
2. Image Segmentation Model: https://github.com/CSAILVision/semantic-segmentation-pytorch
3. Used as a reference for Transfer Learning Models: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

References

[1] bing-image-downloader. https://pypi.org/project/bing-image-downloader. Accessed: 2022-05-08. 2

[2] Drowning facts. https://www.cdc.gov/drowning/facts/index.html. Accessed: 2022-05-08. 1

[3] Facts stats about drowning. https://www.stopdrowningnow.org/drowning-statistics/. Accessed: 2022-05-01. 1

[4] Swimeye. https://swimeye.com/. Accessed: 2022-05-08. 2

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 7

[6] 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. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[8] Nicolas Jacquelin, Romain Vuillemot, and Stefan Duffner. Detecting swimmers in unconstrained videos with few training data. *Machine Learning and Data Mining for Sports Analytics (MLSA)*, 2021. 1, 2

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 2

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[13] Gordon S Smith. Drowning prevention in children: the need for new strategies. *Injury Prevention*, 1(4):216, 1995. 2

[14] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 4

[15] Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13:12, 2019. 6

[16] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 3

[17] Troy W Whitfield. An evaluation of swimming pool alarms. *Cosumer Product*, 2000. 2

[18] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3

[19] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018. 2

[20] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 2