# Strava Route Art Sketch Image Classification

Anthony Xie, Dylan Cunningham
Stanford University
anthonyx@stanford.edu, dcunningham@stanford.edu

## Abstract

*Inspired by image sketch classification, our project attempts to classify Strava route art into ImageNet classes. The Strava route art images are largely sparse and are unique, abstract representations of objects. Studying this problem can help us learn more about the general intelligence of computer vision models based on if they can interpret sketches. We take five models pre-trained on ImageNet and fine-tune these models on ImageNet-Sketch, a large sketch dataset that exhibits similar properties to our Strava art test dataset. All of our pre-trained models had some element of residual connections in their architecture, because we found that to be promising in our research of related work. We saw significant improvement in test set top-1 and top-5 accuracy scores for 3 out of our 5 models after fine-tuning, which demonstrates that those models were effective were able to effectively transfer learn. However, the improved accuracy scores are still low and we were expecting to see improvement on all 5 models so there is clear room in this project for future improvement.*

## 1. Introduction

Strava, a popular fitness app, allows runners, bikers, and other athletes to upload their activities and view the activities of friends. Strava has over 100 million registered athletes and those athletes uploaded 2.5 billion activities over the last 18 months [14]. Often, activities are represented as a map with the route that was traveled along with statistics such as distance and elevation gain. Occasionally, athletes choose to run routes in a specific fashion so that the image will create a drawing of a particular object or animal. As shown in the orange route in Figure 1, an athlete started at Oracle Park in San Francisco and then ran through the streets of the city in a specific order so that their route would look like a drawing of a baseball. The strava community collectively recognizes routes like the baseball route as Strava art and often people on the app will name their routes based on the object their route most clearly resembles. The purpose of this project is to classify Strava art.


Figure 1. Strava art. A map of San Francisco containing an orange route line that resembles a baseball.

Specifically, we are fine-tuning computer vision models to properly classify Strava art into the proper ImageNet class [4]. The input to our model is an image containing a Strava route and the model then outputs a predicted ImageNet class. The goal of this project is to achieve an improved accuracy score on our custom Strava art test dataset compared to our baseline methods. We use a variety of model architectures but have found that models with residual blocks perform best on the task of classifying Strava art.

This problem of Strava art classification can be viewed as an abstraction of the problem of drawing or sketch image classification. Strava art at its core tends to be line outline drawings of objects. Strava images share similarity with other sketches and these qualities make them unique to study when considering image classification - they are highly sparse, they feature levels of abstraction that may differ from artist to artist, and can still be recognizable even if transformed or shown from different angles. Humans are able to quickly sketch simplified drawings that other humans can recognize as representations of objects, animals, or people. There are a small amount of features and detail in these images and yet the human brain can correctly understand meaning based on a simple sketch. One study

showed participants pictures of objects that were professionally photographed or simplified line drawings of the same objects and found that participants had extremely similar reaction times and error rates in identifying the objects in each condition [2]. These results suggest that viewing a simplified sketch of an object is nearly equivalent in humans for the purpose of object recognition.

Therefore, by studying how computer vision architectures perform on drawing or sketch data, we can learn important information about how similar the detection characteristics of computer vision models and human brain are. Furthermore we can gain an understanding on how well the models are truly gaining an understanding of the objects they view. The ability to recognize outline line drawings can give us an idea of the general vision intelligence of these models. By working on the problem of Strava art classification, we are contributing to the field of sketch and drawing classification and helping to solve this problem.

Strava art specifically builds on drawing and sketch problems but poses its own unique difficulty through features specific to how it is created. For example, each route must be constructed by a single line, which adds additional lines where two separate forms might be connected (i.e, if one wanted to draw a sun and horizon, there would need to be an additional line between the circular sun and the horizon line), which adds difficulty to the task. Furthermore, paths are strictly limited to whichever roads that the path is created on, further obfuscating certain shapes by adding additional jagged edges and small turns to curves and lines that might be straight when drawn by a pencil.

Finally, this is a fun problem to solve because Strava artists have a vibrant community on the app and on Instagram, and many people create Strava art to express themselves creatively. This work is a fun way to engage with that creativity.

## 2. Related Work

Strava art classification is tightly connected to traditional image classification. We are following a similar approach to many other computer vision papers by taking models with pre-trained weights and then fine-tuning them on a new dataset that more clearly represents our task. Many papers follow this approach and have seen success in their specific vision related task such as object detection [6], visualizing CNNS [24], or video action classification [3]. Taking an effectively pre-trained model and adapting it to a new, more specific task is a standard and proven technique.

No one has published results online showing their approach to the problem of Strava art image classification, so we are working in a new domain. However, as previously mentioned, this domain closely mirrors the domain of sketch and drawing classification which has plenty of prior work.

Early work in sketch classification work was focused on taking low-level features of the sketches and modeling representations or embedding with bag of feature representations [10]. Similarly, one group used fisher vectors to extract meaningful information about sketches and then fed that info into an SVM for classification [12]. Later researchers started exploring using deep neural networks for sketch classification [23]. Yang and Hospedales used a Convolutional neural network with five convolutional-ReLU layers with pooling layers after the first, second, and fifth convolution. The network ends with two fully connected layers. They evaluated their model on the TU-Berlin Sketch dataset [5] and found the highest accuracy score at the time, reaching near human performance.

SketchNet is another deep convolutional neural network that matched real images with sketches in the TU-Berlin dataset to aid in training [25]. The inputs to SketchNet were three images, the sketch itself, a postive example of a real matching image, and then a negative example of a real but not matching image. With SketchNet, researchers were able to achieve state of the art performance on TU-Berlin.

Singer et al. found that CNN based deep learning models for sketch recognition had similar representations in early and intermediate layers for both sketched and normal images. However, there were differences present in the later layers of the networks which can be fixed by finetuning the later layers of the network [13]. This work helped inform us in how we think about finetuning our pre-trained ImageNet models to work well on sketches.

We picked our base models based on prior work the best performing pre-trained models on ImageNet-Sketch [20] without any finetuning. These were found to primarily be IG-ResNeXt models which were pretrained on Instagram data and later fine tuned on ImageNet data.

## 3. Dataset

As the Strava art classification task has never been tackled before, there does not exist a Strava art dataset. Therefore, we had to get creative in figuring out a way to train a model to perform well on Strava art.

We collected and labeled our own Strava art dataset from scratch using photos available on Strav.art [1]. The images vary in resolution but most are around 1000x1500 pixels. We had to filter through all of the images and only take the ones that clearly fit into a specific ImageNet class. Also, we had to make sure that we didn't have images that were very ambiguous as to which ImageNet class they were a part of. For example, there were plenty of images that were route outlines of dogs, but there are at least 20 different ImageNet classes that correspond to different breeds of dogs. Therefore, we chose to not include the route outlines that were generally dog shaped but were very ambiguous as to which breed of dog they represented. Based on these input

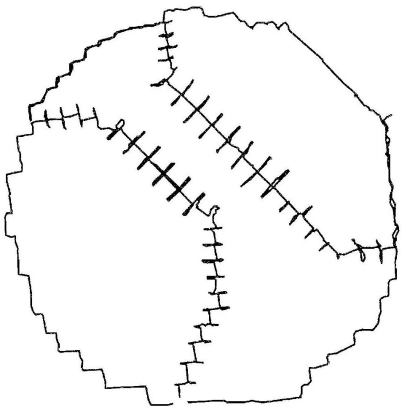

Figure 2. Processed strava art from figure 1 after it has been run through our data processing pipeline. We applied a mask to remove the map from the background and then changed the orange color to black and padded the image to make it square.

factors, we ended up with a Strava art dataset size containing 126 images labeled with one ImageNet class. Because this dataset is so small, we decided to only use it as our test dataset, because we could not realistically expect good results finetuning our models with only 75 examples and validating and testing with 25 each.

We have a few data processing steps to prepare the Strava data for our model. First, we created a mask to remove the map background of the route images so that we could just feed in the outline of the route into the model. Second, we padded the image to make it square so that it wouldn't later be oddly cropped when it was passed into one of the models. Because our Strava art dataset was just being used as a test set, we wanted to make sure that each image was passed into our model in its full form and wasn't oddly cropped or anything. Figure 2 shows an example of a piece of strava art after going through our data processing pipeline.

Since our custom Strava art dataset is so small, we knew we would have to use a different dataset to train our model with. Building on our earlier discussions of sketch datasets, we thought that fine-tuning on a sketch dataset would hopefully improve the model's ability to classify Strava routes. We decided to use the ImageNet-Sketch dataset to fine-tune our pretrained models [19]. ImageNet-Sketch mirrors the original ImageNet [4] dataset but consists of 50,000 sketch images with 50 images for each of the 1000 original ImageNet classes. This dataset was created by pulling sketch images from google images and handpicking the best ones.

Figure 3 shows a few examples of data present in the ImageNet-Sketch dataset. Clearly, some of the images do resemble processed strava route art in the way that they are black and white outlines of objects. However, some of the data samples are much more complicated drawings than what you typically see with Strava art. Still, ImageNet-Sketch tends to have more similar style of images to Strava art than ImageNet itself, so we will be using it to fine-tune.

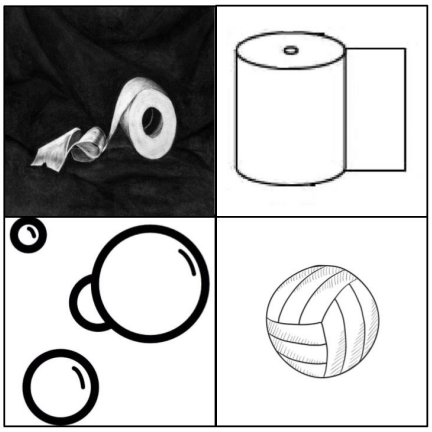

Figure 3. A showcase of a few samples of ImageNet-Sketch data showing the variance in styles

## 4. Methods

At a high level, we are taking five different pre-trained ImageNet models and fine-tuning them with ImageNet-Sketch data and then testing on our Strava test dataset. We evaluated the performance of our pre-trained models on the Strava test dataset to get baseline accuracies. We use transfer learning to fine tune these models to be better suited to our Strava classification task. We elected to use transfer learning instead of completely retraining these models using only the ImageNet-Sketch dataset based on our knowledge of related works within the field. The pre-trained models have been trained on millions of existing ImageNet images, and thus the representations found in their layers will most likely be much more accurate than could be done through our own training. Since the representations found in early and middle layers appear to be similar for both ImageNet and ImageNet-Sketch, we can then use these more accurate representations initially, and then fine-tune the later layers using sketch-based data, which will hopefully help the model generalize and conform to the Strava dataset. As previously noted, because we do not have access to a sizeable enough amount of Strava art to properly fine-tune it, we will be using the ImageNet-Sketch dataset instead. Transfer learning works to leverage the existing pre-trained weights within each model, by freezing certain layers to keep the expressive power behind them, while training other layers in order to match certain new data. The four neural network architectures that we have decided to investigate for use in Strava image classifcation are as follows: ResNet50, EfficientNet, Inception ResNet, and IG-ResNeXt-101. We used the pre-trained versions of all of these models from Pytorch Image Models [21]. We will

first give an overview of all the different model architectures that we are using as our baseline models to fine-tune on top of.

### 4.1. ResNet

ResNet-50 is a deep convolutional neural network that takes advantage of residual connections. ResNets build upon the standard convolutional neural network architecture with the addition of residual blocks. Residual blocks add the result of the previous layer to the result of the next layer after applying a number of convolutions convolutions, then perform the activation of said sum. This allows for learned weights in previous layers to be reused in future layers. [7] The specific layer pattern used in a single residual block can be seen in the following figure.

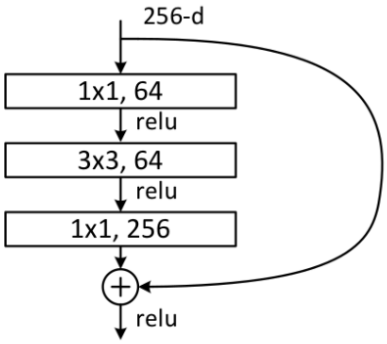


Figure 4. A diagram of the residual block utilized in the residual layers of the resnet-50 architecture

The ResNet-50 architecture is comprised of 5 of these residual blocks, with 50 total layers comprising the entire network, totalling to around 23 million parameters. We elected to choose ResNet-50 in particular due to the fact that it has been highly explored in other transfer learning tasks and has been shown to be quite successful in adapting towards other image classification types, such as for COVID-19 detection. [8].

### 4.2. EfficientNet

With the knowledge that CNNs can become more accuracy when scaled up, EfficientNet scales up models for multiple dimensions – depth, width, and image size. Rather than scale these dimensions arbitrarily, they scale these with a constant ratio, for a compound scaling method with a set of fixed scaling coefficients. It was found that scaling in this way produced more accurate models while still maintaining the same amount of computational needs.

By using this scaling method on neural network architectures found using a search process that identifies the best models that achieve a strong mix of accuracy and latency [17], they were then able to obtain a set of models, called EfficientNets, that both outperform and are more computationally efficient than other ConvNets. We elected to use EfficientNet-B5, as it improves upon the top-1 accuracy of ResNet-50 from 76.3% to 83.0% for ImageNet classification, while keeping the computational needs around the same. EfficientNets have also been shown to perform well for transfer learning, which is crucial for our methodology. [18]

### 4.3. Inception ResNet

Inception ResNet is a deep convolutional neural network that combines both inception modules as well as residual connections described in the ResNet section [15]. Inception modules are meant to reduce computational complexity by incorporating multiple sized filters typically including 1x1 convolutions and concatenating the results together [16]. In Inception ResNet specifically, the filter concatenation stage of Inceptions are replaced with residual connections between layers.

We used Inception-ResNet-v2 which had been pretrained on ImageNet [4]. Inception-ResNet-v2 uses multiple residually modified inception blocks followed by reductions or average pooling layers. We were curious to see how the residual inception module would perform in a task of transfer learning, so that is why we wanted to experiment with this model.

### 4.4. IG-ResNeXt

IG-ResNeXt [11] uses grouped convolution ResNeXt layers. ResNeXt layers use aggregated residual transformations . This is similar to Inception ResNet because you split and later bring together results in the residual function. Unlike Inception-ResNet, ResNeXt uses a shared topology for all of its paths like classic ResNet modules [22]. With ResNeXt modules, you choose what cardinality you want within your modules. Cardinality is a measure of the size of the set of transformations.

We will be specifically using IG-ResNeXt-101 32x8d and IG-ResNeXt-101 32x16d. The only difference is in the cardinality of the ResNeXt blocks, 8 and 16 respectively. The larger the cardinality, the larger the number of parameters. Both models have 101 layers and 32 ResNeXt blocks.

A pre-trained IG-ResNeXt seems to be especially robust for transfer learning because of the datasets it was trained on. First, the model was pre-trained on a huge Instagram hashtag dataset [11] and then later trained on ImageNet [4]. The authors also reached state of the art results on ImageNet-1k benchmark task. Pre-trained IG-ResNeXt models performed best on ImageNet-Sketch without any fine-tuning [20], so we decided to use two different versions to see which performed better.

Table 1. Model top-1 and top-5 accuracy scores on Strava test dataset both before and after fine-tuning with ImageNet-Sketch

| | No Fine-Tuning | | With Fine-Tuning | |
|---|---|---|---|---|
| Model | Top-1 % | Top-5 % | Top-1 % | Top-5 % |
| ResNet50 | 0.00 | 1.59 | 5.55 | 16.66 |
| tf-efficientnet-b5 | 0.79 | 3.18 | 7.93 | 18.25 |
| Inception-ResNet-v2 | 1.59 | 3.97 | 7.14 | 19.84 |
| ig-resnext101-32x8d | 26.98 | 52.38 | **28.57** | 46.03 |
| ig-resnext101-32x16d | **28.57** | **54.76** | 23.01 | 34.92 |

### 4.5. Fine-Tuning Process

We took these fine tuned models and used the fast.ai [9] library's fine-tuning functionality to fine-tune each pre-trained model. The fine-tuning works by initially freezing all layers except the very last to train for a single epoch (the freeze epoch), and then unfreezing all the layers to train for any given number of epochs, at a reduced learning rate as to not lose the information gained in the pretraining and to solely improve the performance of the model. The learning rates for each different model during the fine-tuning process were chosen using fast.ai's functionality, which examines the loss at differing learning rates, and identifies which is likely the best to use for each particular model, as seen in the figure below.

On top of this, we borrowed setup code from the creator of Pytorch Image Models, R. Wightman to get our basic validation loop and test baselines for our models [20]. We wrote code on top of this to fine-tune and later test our models. We also wrote all of our own data-processing code.

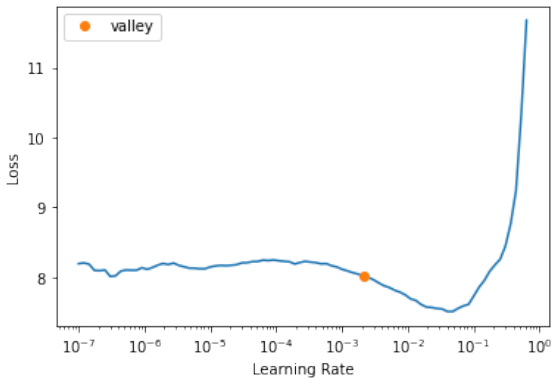


Figure 5. A showcase of learning rate impact on loss, and the identified learning rate used for one of our models

## 5. Experimental Results

The only hyperparameter we edited was the learning rate for fine-tuning and as mentioned previously, we used one of the fast-ai tools to do a learning rate search. For example, on IG-ResNeXt-101 32x8d we used a learning rate of 2e-3. We used a mini-batch size of 24, because it was what was used in the IG-ResNeXt paper.

Our quantitative results are listed in table 1. We measured top-1 accuracy, which was the percentage of images that the corresponding model had a prediction that exactly matched the target label. We also measured top-5 accuracy, which is the percentage of samples where the model correctly predicted the target class in one of its 5 highest weighted class predictions.

## 6. Discussion

Overall we found that for all models besides the IG-ResNeXt models, top-1 and top-5 accuracy improved upon the baseline accuracy after fine-tuning. IG-ResNeXt 8d saw slight improvement with top-1 accuracy scores but slight decline in top-5 accuracy. IG-RexNeXt 16d saw a decline in both top-1 and top-5 accuracy.

Our best two models, the two IG-ResNeXt models seemed to consistently make predictions in the proper domain. For example, when animals were misclassified, they were consistently classified as other animals rather than random objects. For example, a few bald eagle photos were classified as Albatross or Timber wolf was classified as dog. On the other hand, objects tended to be classified as other objects rather than creatures. This showed promise that our best models had a somewhat rich representation of the world.

The two IG-ResNeXt models also tended to perform best on classes which had clearly distinguishable features that were largely unique and not shared within other classes that could be easily represented even using few lines. Classes such as snail contained elements that could be easily identified from multiple angles even when the drawings were incredibly sparse, specifically in this case the spiral pattern of the snail shell which was present in all Strava images.

We found it interesting that one of the most common classes that was classified before fine-tuning was "jigsaw." The outlines of shapes but lack of other content seemed to indicate "jigsaw" to the model but we saw a lot less of this after fine-tuning on ImageNet Sketch.

We found that all models were prone to misclassifying images with a number of closely related other classes. One

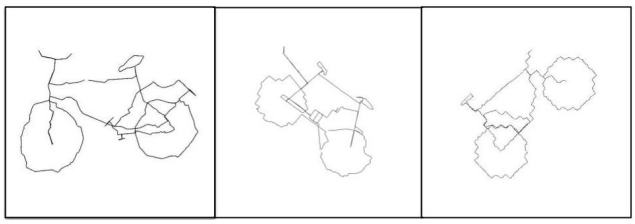

Figure 6. Three "mountain bike" images all misclassified by all of our fine-tuned models as "tricycle"

of the most notable examples was for mountain bikes, as seen in Figure 6, where a number of images in the class were consistently incorrectly labelled as tricycle, and occasionally unicycle and tandem-bicycle.

In addition to this, our models also often struggled with more abstract representations of objects, especially if the image itself was incredibly sparse and low on detail. There were a number of Strava images that were very vague and thus were incorrectly labelled as a wide range of classes, often times being classes that were completely unrelated. An example of a commonly incorrectly labelled image can be seen in figure 7.

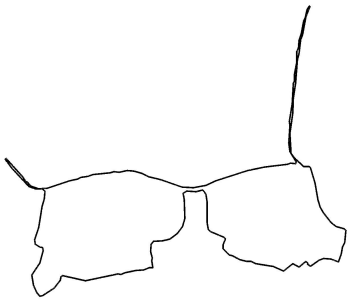

Figure 7. An image labelled "sunglasses", incorrectly labelled as a number of different classes by our models, such as "meat loaf" and "maillot"

In contrast to this, highly detailed Strava images which did not rely on abstract and sparse representations all performed well within our fine-tuned models, which met our expectations since these types of images would be most similar to those found in the ImageNet-Sketch and ImageNet databases. An example of one correctly labelled for all classes can be seen in Figure 8.

## 7. Conclusions and Future Work

We found that taking pre-trained residual based deep convolutional neural networks and finetuning them on sketch data tended to lead to improvement on classifying Strava art. ImageNet-Sketch served as a mostly effective dataset to carry out transfer learning to Strava art, but it was not the most ideal. Moving forward we hope to further improve our classification accuracy by gathering a much larger dataset so that our models can actually be trained on Strava or Strava-like data, rather than fine-tuned on ImageNet-Sketch. This can potentially be crowdsourced by having images drawn of classes in which the image must be limited to a single line or portions of a map. In addition, we would also like to further investigate the fine tuning of the IG-ResNeXt with different hyperparameters, as the learning rate and epoch amount we chose seemed to be less effective when compared to the considerable improvements seen by the other models. Also, if we were to do this project again, we would not use the FastAI fine tuning framework and instead write all of our own training loops so that we could have more freedom to make specific changes during training and track progress. FastAI worked well for the purposes of a project like this, but to see much better improvement we want to write everything ourselves.

Figure 8. "Scuba diver" image, correctly predicted for all fine-tuned models

## 8. Contributions and Acknowledgements

**Dylan Cunningham:** Strava art data collection and labeling, code for data preprocessing and augmentation, sketch classification research, FastAI setup, baseline model accuracy evaluations, project writeup

**Anthony Xie:** Strava art data collection and labeling, CNN model research and decision making, code for model fine-tuning and hyperparameter searching, project writeup

## References


[1] Strava art, https://www.strav.art. 2

[2] Irving Biederman and Ginny Ju. Surface versus edge-based determinants of visual recognition. *Cognitive Psychology*, 20(1):38–64, 1988. 2

[3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1, 3, 4

[5] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012. 2

[6] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4

[8] Md. Belal Hossain, S.M. Hasan Sazzad Iqbal, Md. Monirul Islam, Md. Nasim Akhtar, and Iqbal H. Sarker. Transfer learning with fine-tuned deep cnn resnet50 model for classifying covid-19 from chest x-ray images. *Informatics in Medicine Unlocked*, 30:100916, 2022. 4

[9] Jeremy Howard et al. fastai. https://github.com/fastai/fastai, 2018. 5

[10] Yi Li, Yi-Zhe Song, and Shaogang Gong. Sketch recognition by ensemble matching of structured features. pages 35.1–35.11, 01 2013. 2

[11] Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. *CoRR*, abs/1805.00932, 2018. 4

[12] Rosália G. Schneider and Tinne Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. *ACM Trans. Graph.*, 33(6), nov 2014. 2

[13] Johannes J. D. Singer, Katja Seeliger, Tim C. Kietzmann, and Martin N. Hebart. From photos to sketches - how humans and deep neural networks process objects across different levels of visual abstraction. *Journal of Vision*, 22(2):4–4, 02 2022. 2

[14] Strava. Strava's global community continues strong growth surpassing 100m registered athletes on the platform. *Cision PR Newswire*, 2022. 1

[15] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. 4

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. 4

[17] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. 2018. 4

[18] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. 4

[19] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019. 3

[20] Ross Wightman. Generalization to imagenet-sketch. https://www.kaggle.com/code/rwightman/generalization-to-imagenet-sketch/notebook. 2, 4, 5

[21] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 3

[22] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. 4

[23] Yongxin Yang and Timothy M Hospedales. Deep neural networks for sketch recognition. *arXiv preprint arXiv:1501.07873*, 1(2):3, 2015. 2

[24] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. 2

[25] Hua Zhang, Si Liu, Changqing Zhang, Wenqi Ren, Rui Wang, and Xiaochun Cao. Sketchnet: Sketch classification with web images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2