# Generalizing Pretrained Image Classifiers for Free-hand Sketch Recognition

Micah Cheng
University of Michigan
Ann Arbor, MI 48109
mwcheng@umich.edu

## Abstract

*Sketches are a simple and useful way to convey information without a large amount of effort, allowing ideas to visually flow across language barriers. Relative to their natural image counterparts, free-hand sketches are typically more abstract, less detailed, and deformed, with differing intraclass features based on the artist's style and artistic ability. We will be comparing the use of generalizing pre-trained ImageNet models and fine-tuning them for free-hand sketch classification. In our experiments we fine-tuned eleven different types of pretrained ImageNet models. We show that some recent deep neural network architectures trained on natural images can be better than some architectures that are specifically designed for sketch recognition. We also propose that the commonly used AlexNet performs sub optimally on sketch recognition, and that residual networks are a better alternative. Using the TU-Berlin benchmark from Eitz et al. [1] we were able to achieve a top-1 accuracy of 77.7% and top-5 accuracy of 94.5% with the ResNeXt architecture [28].*

## 1. Introduction

Sketching is one of the first methods of written communication, able to convey concepts without much effort and across languages. They are often simpler than their natural image counterparts, emphasizing certain features and ignoring other characteristics, as humans easily recognize objects from schemas consisting of basic shapes and features. They can be used powerfully in a variety of ways, making sketch recognition an interesting and important area of study in computer vision.

Sketch recognition is similar to the well-studied area of image recognition. Nonetheless, sketch recognition is more difficult, as sketches differ from images in two significant ways. First, images and sketches both have intra class variation, showing differences in camera angle, lighting, and perspective. However sketches can also vary based on an artist's style, interpretations, and ability, which are not necessarily grounded in the real world. Shapes and proportions can be distorted, with simplified, missing, or caricaturized parts (such as stick figure limbs or bigger ears on an elephant). Secondly, sketches are often less detailed with less context, and lack color, lighting, and texture information. This makes it more difficult to distinguish between similarly shaped objects, or understand depth and relative size.

In this paper, we generalize several pre-trained ImageNet models for sketch recognition classification. We focus specifically on free-hand sketches by non-artists, and demonstrate how pretrained ImageNet models can be fine-tuned for sketch recognition. Current Convolutional Networks (CNNs) for sketch classification use Sketch-a-Net [10] (which is based off AlexNet), AlexNet [18], ResNet [21], or some modified version of the listed architectures. However there are many newer backbones that have shown improved performance in image recognition. Zheng et al. [2] compared additional architectures SqueezeNet [20], Inception V3 [27], VGG [19], ResNet [21], and DenseNet [22]. Xu et al. [3] compared recurrent neural networks (RNN), graph neural networks (GNN), and CNN baselines in their benchmark. For CNN, they looked at AlexNet [18], VGG [19], Inception V3 [27], ResNet [21], DenseNet [22], and MobileNet [24]. While ResNeXt [28] and Wide ResNet [29] are types of ResNets, these specific architectures as well as MNASNet [26], to the best of our knowledge, have not been used in benchmark comparisons or sketch classifiers.

## 2. Related Work

The development of SketchPad [4] initiated study into the area of sketch recognition. However it was not until the publication of a large free-hand sketch database, Eitz et al. [1], that this field has gained more attention. Sketch recognition has generally followed the related and more studied field of natural image computer vision, and used similar methods to continue to improve sketch recognition.

Like image recognition, early sketch recognition used hand-crafted features, extracted and fed into a conventional classifier. Initial methods employed traditional features used for images, such as Bag of Words (BoW) [5], HOG

| Year | Model | Type | Architecture | Input | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| 2015 | Sketch-a-Net [10] | CNN | AlexNet-like | picture | TU-Berlin | 0.7490 |
| 2015 | DeepSketch [11] | CNN | Sketch-A-Net and AlexNet | picture | TU-Berlin | 0.7542 |
| 2016 | AlexNet-FC-GRU [15] | CNN-to-RNN cascaded | AlexNet | stroke accu. pic. | TU-Berlin 160 cat. | 0.8510 |
| 2017 | DeepSketch3 [13] | CNN | Sketch-A-Net and AlexNet | picture | TU-Berlin | 0.7918 |
| 2017 | Jia et al. [32] | RNN-RNN dual branch | Sketch-A-Net | CNN features of stroke accu. pic. | TU-Berlin | 0.9220 |
| 2017 | DVSF [33] | R-FC and RNN dual branch | - | CNN features of stroke accu. pic. | TU-Berlin | 0.7960 |
| 2018 | SketchMate [31] | CNN-RNN dual-branch | AlexNet | picture & stroke vector | QuickDraw 3.8M | 0.7949 |
| 2018 | R2CNN [35] | RNN-to-CNN cascaded | ResNet-50 | stroke vector | TU-Berlin | 0.7849 |
| 2018 | FBin DAB-NET [34] | binary CNN | Sketch-A-Net, ResNet-18 and GoogleNet | picture | TU-Berlin | 0.7370 |
| 2019 | SSDA [2] | CNN | - | picture | TU-Berlin | 0.8427 |
| 2019 | multi-graph tran [17] | GNN | - | stroke vector | QuickDraw subset | 0.7070 |

Table 1: Comparison of deep learning sketch recognition networks. "-" means it is vague, not mentioned in the original paper, or not applicable. Sketch-a-Net and AlexNet have similar structures. The reported performances are the reported top-1 accuracy. "stroke accu. pic." abbreviates stroke accumulated pictures

[6], and SIFT [7]. Eitz et al. [1], Schneider et al. [7], and Li et al. [8] all used a Support Vector Machine (SVM), and only differed in the features they choose to extract.

Deep learning emerged as a leading framework for sketch recognition, having demonstrated the success of Convolutional Neural Networks (CNN) in natural image recognition challenges such as ILSVRC [9]. The advantage of deep learning, when compared with SVMs, lies in the ability to learn relevant features of an image itself, without relying on hand-crafted feature extraction. Yu et al. [10] introduced the first CNN Sketch-a-Net with an AlexNet-like architecture. DeepSketch [11] and its following work DeepSketch2 [12] and DeepSketch3 [13] soon followed with improved results, also using an architecture similar to AlexNet.

Deep learning requires large datasets to achieve optimal performance, but for early models, very few large-scale sketch databases existed. Even Eitz et al. [1] with 20,000 images over 250 classes is small when compared to the size of ImageNet and other natural image databases. In 2017, Google released QuickDraw [14], a dataset with over 50M sketches from people worldwide. Following this, sketch classification methods became more specialized away from traditional image recognition.

One advantage of sketches, is that they are drawn in a specific stroke order. Treating sketches as an ordered sequence, Recurrent Neural Networks (RNNs) can be used to evaluate this sequential information, as Ha et al. [14] demonstrated in SketchRNN, an RNN-based Variational AutoEncoder (VAE). Further work continued combining CNN and RNN as proposed by Sarvadevabhatla et al. [15].

Recently, sketch recognition has shifted towards using Graph Neural Networks (GNNs) which GNNs iteratively build graphs representing neighboring features. These GNN look at images as a geometric representation, as opposed to seeing the entire image as a set of pixels in Euclidean space to be fed into a CNN. Using various novel methods, such as sketch hashing [16] or sketch transformers [17], GNN methods such as Xu et al. [3] have been successful in achieving similar accuracies to state-of-the-art CNN and RNN models.

Despite a majority of CNN sketch classifiers using a AlexNet or Sketch-a-Net architecture, the design of the backbone in deep neural networks remains an open question. This is evidenced by the many proposed methodologies and architectures within the past five years, including several RNN and CNN architectures which perform at state-of-the-art, including ResNet [21], DenseNet [22], Inception V3 [23], and MobileNet [24].

## 3. Method

In this project we explore eleven different convolutional network architectures. We cannot show each specific architecture in this paper, but we use pretrained ImageNet models from pytorch, replacing the last classification layer to output to 250 classes as opposed to 1000 ImageNet classes. We can categorize the models used into 3 categories with novel methods to increase depth and make the models more accurate and precise.

## 3.1. Classical Networks

The first set of networks are "classical" feed forward convolutional networks, which contain multiple convolutional layers followed by fully connected layers, and only differ in the size of filters applied, layers, and depth. These traditional CNN use pooling to combine outputs from a previous layer to the current layer through a convolutional operation, and include models like AlexNet [18], VGG [19], SqueezeNet [20], MobileNet [24], and MNASNet [26]. AlexNet was an earlier pioneer in CNN backbones and is one of the most common architectures used in CNN sketch recognition. It contained 5 convolutional layers, some of which were followed by max-pooling layers. These convolutional layers were followed by three fully connected layers and used the ReLU activation function.

Some modifications to AlexNet can be made to increase the depth and decrease complexity of it. VGG proposed using smaller convolutions to increase the depth of the network [19]. SqueezeNet showed that by replacing 3x3 filters with 1x1 filters, decreasing input channels to 3x3, and down sampling late in the network, the number of parameters can be decreased by 50x and compressed to 510x smaller while keeping the same accuracy rates [20].

MobileNet and MNASNet are mobile nets, designed to meet certain constraints to be run on mobile devices. They are designed to be small and fast, through the use of dense blocks, which contain a batch normalization, ReLU activation, and convolutional layer.

In CNN, typically greater accuracy can come from an increased depth or width. However, with sketch recognition, deeper networks do not necessarily produce better results. Additionally, increasing depth can overfit the data, an issue with sketch recognition due to limited datasets, and increase computational complexity [30].

## 3.2. Inception Networks

Inception Networks such as GoogLeNet (Inception V1), Inception V2, and Inception V3, build upon classical feed forward convolutional networks [27]. Instead of a deeper network, inception networks are designed to go "wider", through the use of inception modules. Inception modules perform multiple convolutions per layer, consisting of a 1x1, 3x3, and 5x5 filter. Following these, max pooling occurs and the outputs are concatenated together for the next inception layer. In later inceptions versions, these filters are factorized down to reduce computational expense, from a 5x5 to two 3x3 convolutions, and from a 3x3 to a 3x1 and 1x3 filter. Inception V3 is not used in this paper as it introduces auxiliary classifiers, and the pretrained model assumes input is 299x299px, rather than 224x224px.

## 3.3. Residual Networks

In contrast with classical "plain" nets, residual networks are able to decrease error rate as depth is added through the use of skip connections. Skip connections allow for identity mappings, a layer that does not do anything and simply passes the input to the next layer. Residual Networks use an additive method to learn the residual mapping, denoted as $F(x) = H_l(x_{l-1}) - x_{l-1}$ where $l$ is the current layer. We can then learn the original mapping recast as $F(x) + x_{l-1} = H_l(x_{l-1})$ to achieve a ResNet [21]. Through these "highway networks", ResNets are able to train more efficiently and without a vanishing gradient.

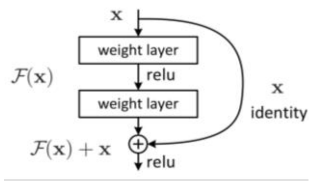
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 1. Building block of a residual unit [21].

ResNet has two common variants, Wide ResNet [29] and ResNeXt [28], which attempt to decrease computational complexity by increasing the dimensions of each layer. First, an alternative to deep networks is adding width, as used in Wide ResNet. By increasing the number of filters per layer, the Wide ResNet can decrease the depth and increase the width of each layer, allowing for fewer parameters when compared to the same depth ResNet. Secondly, ResNeXt [28] adds the idea of another dimension "cardinality" in additional to width and depth. Each ResNeXt block will split into multiple diverging paths, where the number of paths is defined by the hyperparameter "cardinality". Each path follows the typical ResNet structure, and the results are merged together before being passed into the next layer. ResNeXt follows the same intuition as inception networks, performing multiple convolutions in the same layer allowing ResNeXt to go even wider in the additional dimension. However ResNeXt differs in that only a single hyperparameter needs to be tweaked, and outputs are added as opposed to concatenation.

DenseNets [22] also make use of residual blocks, the only difference to ResNets being the use of concatenation between layers as opposed to an additive method. Each of the $l^{th}$ layer receives the feature map of all preceding layers, defined as $H_l([x_0, x_1, \ldots, x_{l-1}])$, where $[x_0, x_1, \ldots, x_{l-1}]$ is the concatenation of feature maps produced in layers $0, \ldots, l-1$.

This dense connectivity allows layers within the same block to share collective knowledge, making learned features non-redundant with this common knowledge.

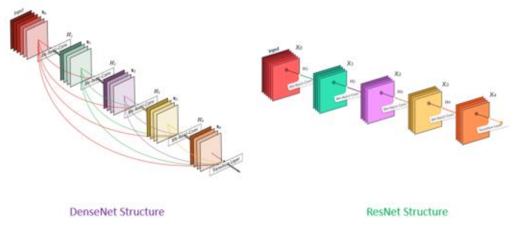

Figure 2. Differences between DenseNet and ResNet [21]. DenseNets receive features maps from all previous layers as input [22].

## 4. Experiments

### 4.1. Dataset

We will be using the sketch database Eitz et al. [1], which contains 80 images per each of the 250 classes, totaling 20,000 images. Each image is given as an 1111x1111px PNG file, but accounting for computational constraints, each image was resized to 224x224px. The dataset was divided 60/20/20 for training, validation, and testing.

### 4.2. Comparative results of different CNN models

We fine-tuned pretrained ImageNet models as provided by pytorch. Each network trained over 20 epochs with a learning rate of 0.001 and the weights with the highest validation accuracy was used to compute test accuracy.

| Architecture | Recognition Accuracy | | Parameter Amount |
|---|---|---|---|
| | Top-1 | Top-5 | |
| AlexNet [18] | 0.659 | 0.880 | 58,028,090 |
| SqueezeNet-1.0 [20] | 0.659 | 0.879 | 863,674 |
| VGG-11_bn [19] | 0.724 | 0.909 | 129,796,090 |
| VGG-11 [19] | 0.697 | 0.891 | 129,790,586 |
| VGG-19_bn [19] | 0.738 | 0.924 | 140,605,498 |
| VGG-19 [19] | 0.721 | 0.905 | 140,594,490 |
| ShuffleNet [25] | 0.168 | 0.392 | 1,509,854 |
| MNASNet [26] | 0.720 | 0.920 | 3,422,562 |
| MobileNet V2 [24] | 0.729 | 0.914 | 2,544,122 |
| GoogLeNet [27] | 0.7175 | 0.918 | 6,624,904 |
| ResNet-18 [21] | 0.735 | 0.923 | 11,304,762 |
| ResNet-101 [21] | 0.757 | 0.930 | 43,012,410 |
| ResNet-152 [21] | 0.7655 | 0.935 | 58,656,058 |
| Wide ResNet-50 [29] | 0.765 | 0.935 | 67,346,490 |
| Wide ResNet-101 [29] | 0.766 | 0.942 | 125,349,946 |
| ResNeXt-50 [28] | 0.755 | 0.940 | 23,492,154 |
| ResNeXt-101 [28] | **0.777** | **0.945** | 87,254,586 |
| DenseNet-121 [22] | 0.751 | 0.933 | 7,210,106 |
| DenseNet-161 [22] | 0.770 | 0.937 | 27,024,250 |
| DenseNet-201 [22] | 0.771 | 0.940 | 18,573,178 |

Table 2. Test accuracies for each of the pretrained models.

Table 2 presents the performance of each CNN architecture on the TU-Berlin dataset.

The residual network variations, ResNet, ResNeXt, Wide ResNet, and DenseNet, are the best performing CNN architectures, with ResNeXt-101 achieving the highest accuracy of 77.7%. Due to time constraints, we do not compare each variation with the same depth. However, we can see between ResNet variations with 101 layers, a simple ResNet falls behind performance from Wide ResNet and ResNeXt.

Secondary to residual networks, we see that VGG, GoogLeNet, MobileNet, and MNASNet all performed at the same level. This shows that deeper networks (VGG) perform at the same level as shallow networks (GoogLeNet, MobileNet V2, and MNASNet). This result highlights that CNNs designed for images filled with colors and textures are not always optimal for sparse free-hand sketches.

We also see that AlexNet performs substantially worse than all other methods. It is a much more shallow and simple architecture, indicating a minimum level of complexity to extract abstract features from sketches.

### 4.3. Results comparisons

Of the models, ResNeXt-101 performed the best with a test accuracy of 77.7%. This is significantly better than the initial Eitz et al. [1] benchmark of 56%. We also performed better than human (73.1%) [1] and Sketch-a-Net (75.9%). However we see that simply fine tuning a model is not able to achieve state-of-the-art accuracies.

## 5. Conclusion

In this paper we explore other possible architectures for CNNs in sketch recognition. Deep learning in sketch recognition is new and most choose to use AlexNet [18] as a simple and common backbone. In fact only Zheng et al. [2] and Xu et al. [3] to our knowledge have done major baseline comparisons with other architectures. However from our results we demonstrate that AlexNet performs substantially worse, despite being the choice of architecture in a majority of CNN sketch classifiers. We show that deeper networks also perform better to a certain extent, and that wide networks such as Wide ResNet [29] or ResNeXt [28] can compensate for depth. Due to limited time we were not able to explore then effects of parameters on each model.

Additionally, the QuickDraw [14] or a subset of it has yet to become widely used in the sketch recognition community, as there currently is not cleaned dataset. However the commonly used and much smaller TU-Berlin dataset limits deep learning for sketch recognition, and some architectures may do better when trained on a larger dataset. All source code and models can be found at: https://github.com/mwcheng21/sketch-classification

## References

[1] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? ACM Trans. Graph. (Proc. SIGGRAPH), 31(4):44:1– 44:10, 2012.

[2] Y. Zheng, H. Yao, X. Sun, S. Zhang, S. Zhao, and F. Porikli, “Sketch-specific data augmentation for freehand sketch recognition,” arXiv preprint arXiv:1910.06038, 2019.

[3] P. Xu, C. K. Joshi, and X. Bresson, “Multi-graph transformer for free-hand sketch recognition,” arXiv preprint arXiv:1912.11258, 2019.

[4] I. E. Sutherland. Sketchpad a man-machine graphical communication system. Transactions of the Society for Computer Simulation, 2(5):R–3, 1964.

[5] R. Hu, T. Wang, and J. Collomosse. A bag of-regions approach to sketch-based image retrieval. In ICIP. IEEE, 2011. 2

[6] R. Hu and J. Collomosse. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. CVIU, 2013. 2

[7] R. G. Schneider and T. Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. ACM Transactions on Graphics (TOG), 33(6):174, 2014.

[8] Y. Li, T. M. Hospedales, Y.-Z. Song, S. Gong, Free-hand sketch recognition by multi-kernel feature learning, Computer Vision and Image Understanding 137 (2015) 1–11.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.

[10] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales. Sketch-a-net: A deep neural network that beats humans. International Journal of Computer Vision, pages 1–15, 2016

[11] O. Seddati, S. Dupont, and S. Mahmoudi. Deepsketch: deep convolutional neural networks for sketch recognition and similarity search. In Content-Based Multimedia Indexing (CBMI), 2015 13th International Workshop on, pages 1–6. IEEE, 2015.

[12] O. Seddati, S. Dupont, and S. Mahmoudi. Deepsketch 2: Deep convolutional neural networks for partial sketch recognition. In Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on, pages 1–6. IEEE, 2016.

[13] O. Seddati, S. Dupont, and S. Mahmoudi (2017) DeepSketch 3 analyzing deep neural networks features for better sketch recognition and sketch-based image retrieval. Multimed Tools Appl 76(21):22333–22359

[14] D. Ha and D. Eck. A neural representation of sketch drawings. In ICLR, 2018

[15] R. K. Sarvadevabhatla, J. Kundu, and V. Babu R, Enabling my robot to play pictionary: Recurrent neural networks for sketch recognition. In ACM MM, 2016.

[16] P. Xu, Y. Huang, T. Yuan, K. Pang, Y-Z. Song, T. Xiang, and T. Hospedales. Sketchmate: Deep hashing for million-scale human sketch retrieval. In Proc. CVPR, 2018.

[17] P. Xu, C. K. Joshi, and X. Bresson, “Multi-graph transformer for free-hand sketch recognition,” arXiv preprint arXiv:1912.11258, 2019.

[18] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Neural Information Processing Systems, 2012, pp. 1097–1105.

[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for largescale image recognition, arXiv:1409.1556.

[20] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size, arXiv:1602.07360.

[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[22] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2261–2269.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. CVPR, 2018.

[25] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design. ECCV, 2018.

[26] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, MnasNet: Platform-aware neural architecture search for mobile. CVPR, 2019.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.

[28] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated ´ residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431, 2016.

[29] S. Zagoruyko and N. Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.

[30] P. Xu, “Deep learning for free-hand sketch: A survey,” arXiv preprint arXiv:2001.02600, 2020

[31] P. Xu, Y. Huang, T. Yuan, K. Pang, Y.-Z. Song, T. Xiang, T. M. Hospedales, Z. Ma, and J. Guo, “Sketchmate: Deep hashing for million-scale human sketch retrieval,” in CVPR, 2018.

[32] Q. Jia, M. Yu, X. Fan, and H. Li, “Sequential dual deep learning with shape and texture features for sketch recognition,” arXiv preprint arXiv:1708.02716, 2017

[33] J.-Y. He, X. Wu, Y.-G. Jiang, B. Zhao, and Q. Peng, “Sketch recognition with deep visual-sequential fusion model,” in MM, 2017.

[34] A. Prabhu, V. Batchu, S. A. Munagala, R. Gajawada, and A. Namboodiri, “Distribution-aware binarization of neural networks for sketch recognition,” in WACV, 2018.

[35] L. Li, C. Zou, Y. Zheng, Q. Su, H. Fu, and C.-L. Tai, “Sketchr2cnn: An attentive network for vector sketch recognition,” arXiv preprint arXiv:1811.08170, 2018.