Free-hand Sketch Classification
CNN, RNN and Dual Branch Attention

Zhenyu Tang*, Ziyin Zhang*, Zirui Liu*
Shanghai Jiao Tong University
Shanghai, China
{tang_zhenyu, daenerystargaryen, l.prime}@sjtu.edu.cn

Abstract

Sketches with sparse strokes rather than dense pixels are highly abstract and practically convenient for humans, yet they pose many challenges to image recognition tasks. In this project, we conduct experiments on QuickDraw sketch dataset using both convolutional neural networks and recursive neural networks, compare the performance of several well-performing network architectures, analyze their strengths and weaknesses, and make our own improvements. Among the convolutional networks, EfficientNet achieved a highest test score of ***83.93%****, and with the help of Long Short-Term Memory it pushed the score even higher, to* ***84.97%*** *with CNN and RNN in dual branch and a branch attention for fusion.*

1. Introduction

Free-hand sketch drawing is a common way of conveying messages and expressing emotions among humans and other primate animals. As sketches embed vivid categorical features of the target objects and distinct visual appearances in simple manipulation of strokes, they are extremely valuable to understanding the process of human cognition. The recent development of hand-held devices and multi-modal searching techniques also gave rise to the demand for various sketch tasks including sketch recognition, sketch retrieval and hashing, sketch-photo retrieval and generation, sketch generation, sketch grouping, sketch parsing, sketch segmentation, sketch simplification and abstraction and so on.

As sketches are expressed by stokes, leaving most space of the canvas blank and the generated image with sparse signals, they pose extra challenges over the classical image classification problem. In general, there are three families of methods to process sketches, namely Convolutional Neural Networks(CNNs), Recurrent Neural Networks(RNNs) and Graph Neural Networks(GNNs). In this project, we explore the application of commonly known image classification methods, majorly CNN, on QuickDraw [3] sketch dataset, as well as some variants leveraging the unique properties of sketches with RNN and other techniques, such as pretraining.

In our experiments, among the purely CNN-based models EfficientNet [19] outperformed the others and achieved a test accuracy of 83.93% initialized with pretrained weights, while purely RNN-based models barely reached an accuracy of 57% due to the high-level abstraction of strokes. However, the hybrid CNN-RNN dual-branch model's performance surpassed CNN models, and reached 84.49% with simple fusion of the two branches' decision values, and 84.97% using a multilayer perceptron (MLP) to learn a branch-level attention.

The rest of this work is organized as follows. In Section 2, we briefly review the related works in the research community concerning sketch recognition and its two foundations - image classification and sequence classification. Then, in Section 3 and Section 4 respectively we discuss our work in using CNN-based approaches and RNN-assisted approaches to tackle the task of sketch recognition. In each section, we first give the experimental setting, then discuss our methods and analyze the empirical results. Lastly, we discuss the limitations of our work in Section 5 and draw our conclusions in Section 6.

2. Related Work

2.1. Image Classification

Image classification has long been a popular research topic in machine learning. In the past few decades, con-

*Our project source code is available at https://github.com/Vladimirovich2019/CS420-Project.

volutional neural networks have been the most widely adopted approach for image classification, as the convolution operation can detect local spatial structures with a very small amount of parameters compared with fully connected layers.

In 1998, LeNet et al. [9] applied a 7-layer CNN to recognize hand-written digits, pioneering the era of convolution in machine learning. Benefited from the development in computation devices, AlexNet [8], an eight-layer convolutional neural network with 60 million parameters and trained on multiple Graphics Processing Units (GPUs) was proposed in 2012, winning the ImageNet Large Scale Visual Recognition Competition (ILSVRC) of that year and setting off the fetish for deep learning. Since then, countless models and variants have been proposed, some of the most notable ones being Inception [18] which concatenates the features maps of different-sized convolutional kernels (including 1×1 kernel) to achieve a deeper network architecture, VGG which uses 3×3 small kernels, and ResNet [4] which introduces skip connections into the network to tackle the difficulty of back propagation in deep models, extending the network architecture to a staggering number of 152 hidden layers and becoming the de facto default model for image classification since.

Since the proposal of ResNet, convolutional networks have been continuingly scaling up free from the concern of vanishing gradient, and there has been a trend toward larger and deeper networks using ResNet-styled skip connections, which often yield better performance. Tan et al. [19] reflected on model scaling for convolutional networks and presented EfficientNets, which utilized the technique of neural architecture search and significantly improved model performance by carefully balancing network depth, width, and resolution.

Recently attention mechanisms [6, 10, 11] with dynamic selection to disregard irrelevant parts have drawn great attention. Transformer [20], an attention-based neural network architecture that is distinctly different from the typical CNN or RNN architecture and originally proposed for natural language processing (NLP) tasks, has also been introduced into computer vision [1] and outperformed various CNN-based methods on most common benchmarks [2].

2.2. Sequence Classification

Unlike images, sequential data - such as text or the stroke representation of sketches - cannot be processed directly by fully connected or convolutional networks, as the length of sequence usually varies between training instances. Instead, they are typically processed by recursive neural networks, which take the hidden states of last time step as input along with the data at next step. One of the most popular RNN architectures is Long Short-Term Memory (LSTM) [5], which deals with the vanishing gradient that RNN usually suffers by introducing several gates to the network to allow for long-range information dependency, as demonstrated in Figure 1[1], where

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_y] + b_f), \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{3}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \tag{4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{5}$$

$$h_t = o_t * \tanh(C_t). \tag{6}$$

However, the limited temporal extent of LSTM restricts the learning of structural complexity of sketches that may be accommodated in sequence embeddings. This shortcoming has been addressed through the emergence of Transformer[20] networks in which slot masking enhances the ability to learn longer term temporal structure in the stroke sequence.

In the past decade, the advent of word embeddings [13, 14, 15] have introduced novel encoding methods into sequence-based tasks. Recently, various methods of attention including spatial attention, channel attention, branch attention and temporal attention have been proposed. Among them the spatial attention-based Transformer [20] is one of the most groundbreaking works.

2.3. Sketch Recognition

For the task of sketch recognition, typical methods view sketch information as either 2D images or 1D sequences. Yu et al. [25] proposed Sketch-a-Net with AlexNet [8] as backbone learning from the sketch image and for the first time beat human performance. It was also the first deep CNN designed for free-hand sketch tasks. Ha et al. [3] presented an RNN-based network for stroke sequence encoding and decoding, together with a Variational Bayesian Autoencoder (VAE) framework learning a generative model for free-hand sketches.

Sketches can also be recognized with the hybrid of image-based methods and sequential methods. The combining of the two methods can either be dual branches [7, 24, 21] in parallel or a cascaded architecture [12, 17]. The two ideas differ in the fusion step: dual-branch methods conduct late fusion on features

[1]Figure from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

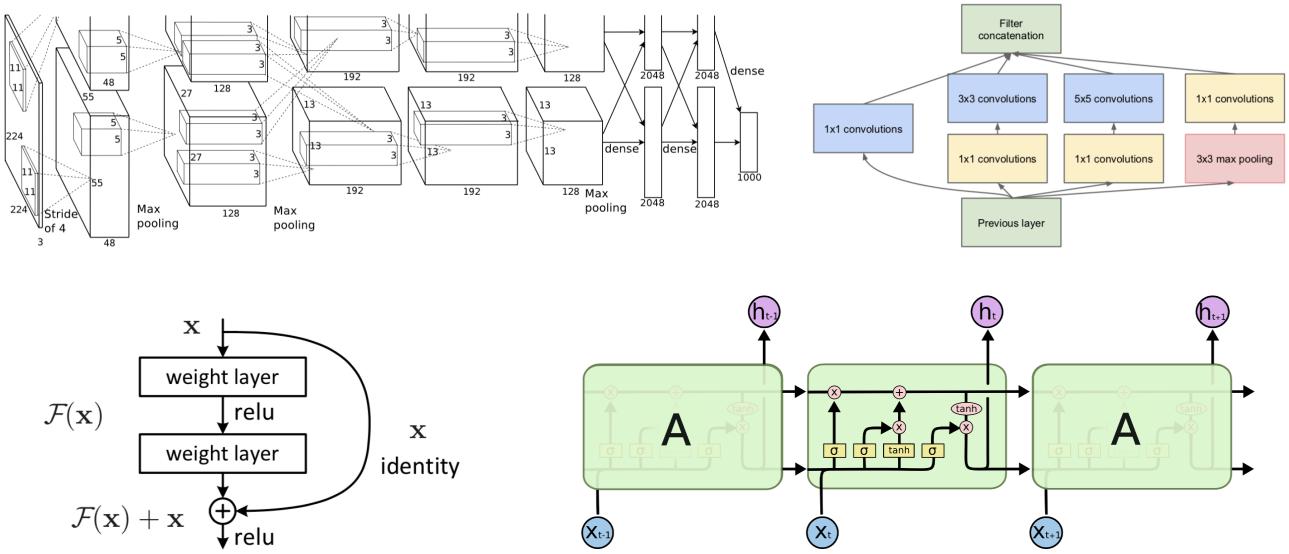


Figure 1: The architecture of AlexNet (top left), and the network blocks of Inception (top right), ResNet (bottom left), and LSTM (bottom right)

extracted by CNN and RNN, while cascaded methods use early fusion.

More specifically, the paper SketchMate [24] introduces a novel problem of sketch hashing retrieval (SHR) and a multi-branch CNN-RNN architecture that specifically encodes the temporal ordering information of sketches to learn a more fine-grained feature representation. The authors combine RNN stroke modeling with conventional CNN under a dual-branch setting to learn better sketch feature representations. However, the problem of visual abstraction, especially how it can be accommodated under a large-scale retrieval setting remains unsolved.

Inheriting the spirit of SketchMate, the paper of sketch-R2CNN [12] proposes a novel end-to-end single-branch network architecture RNN-Rasterization-CNN (Sketch-R2CNN for short) to fully leverage the vector format of sketches for recognition. Sketch-R2CNN takes a vector sketch as input and uses an RNN for extracting per-point features in the vector space. The authors then develop a neural line rasterization module to convert the vector sketch and the per-point features to multi-channel point feature maps, which are subsequently fed to a CNN for extracting convolutional features in the pixel space. The neural line rasterization module is designed in a differentiable way for end-to-end learning.

However, CNNs and RNNs focus on local features, while human beings recognize sketches by global shapes. To utilize the global shape information, Zhang et al. [26] proposed a dual-branch hybrid CNN and used a shape net for extracting global shape descriptors. More recently, GNN-based methods are also proposed for sketch recognition. Xu et al. [23] proposed Multi-graph Transformer to preserve geometric information of strokes and resolve the drawbacks of sketch modeling in Euclidean space in models that are based solely on either images or sequences, yet the performance was not as good as CNNs.

3. Classification with CNN

3.1. Experimental Settings

As a baseline, we first treat the sketch classification problem as a general image classification task, and apply several convolutional neural network to it, like ResNet, MobileNet, EfficientNet along with their different versions.

The dataset we use is a subset of QuickDraw, which includes 25 categories of animal sketches, each with 75 thousand training samples, 2500 validation samples, and 2500 test samples. This amounts to a training set of 1.9 million samples, and a validation and test set both of 62.5 thousand samples.

As the raw sketch data of QuickDraw are stored as stroke sequences rather than RGB images, we first transform the data to make them fit for convolutional operations. The sequences have three channels with various lengths, where the first two channels are the strokes' relative offsets at each step with respect to the previous step on axes x and y, and the third channel is binary bits indicating the state of the pen, i.e. whether

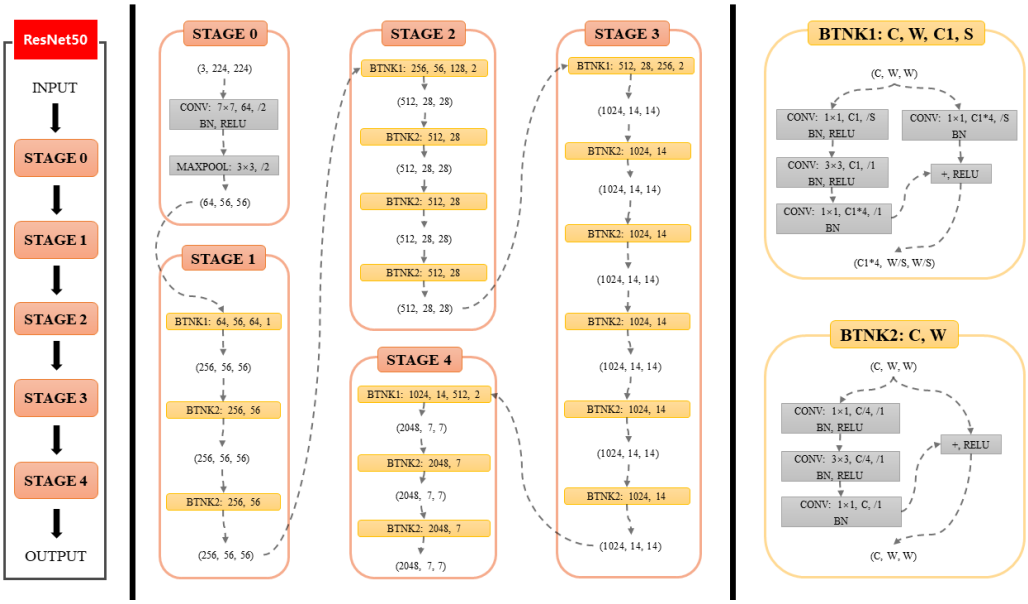


Figure 2: The structure of ResNet-50

it is touching the paper of lifted at this step. Using this information, we translate the strokes of the sketches and store them as 28×28 standard RGB images.

To measure the performance of the sketch recognition models, the research community typically use top-k accuracy [22]. Top-k accuracy is computed as the number of labels in the top k predictions for each instance of the model divided by total number of samples. Researchers often use 1, 3, 5, 10 etc for k, while in our experiment we use top-1 accuracy since the number of classes is relatively small (being 25).

3.2. Methods

In this task, we make use of several effective network structures in our training, including ResNet50, ResNet101, MobileNet-v2 [16], EfficientNet-b0, EfficientNet-b7 [19]. As we mentioned in Section 2.1, ResNet has almost become the default model for image classification since its advent. In this project, we first compare the performance of several ResNet-based models, and make some improvements on them.

The key idea of ResNet is the application of skip connections (figure 1), where in each network block the input undergoes an identity map (skip connection) and is added to the normal output of linear transformation and Rectified Linear Unit (ReLU) nonlinear activation. In this way, in the backward pass of training the gradient is back propagated through both the ReLU connection and the identity connection and prevented from vanishing, solving the problem that almost all deep neural networks had suffered before ResNet. In this work, we use two different configurations of ResNet: ResNet-50 (about 25 million parameters) and ResNet-101 (about 45 million parameters). The network structure for ResNet-50 can be referred to Figure 2[2]

We also compare the performance with several other convolutional networks, including MobileNet [16], which is based on inverted residual structures and has about 3.4 million parameters, being especially designed for lightweight tasks and fast inferences. The model structures can be referred to Figure 4[3]. We also make use of EfficientNet [19], the currently state-of-the-art convolutional network with a very efficient network architecture. Its components and the best performing type Efficient-b7's structure can be referred to Figure 5 and Figure 6[4]. EfficientNet proposes compound model scaling, which combines three types of network scaling: depth, width and resolution, and scales them in accordance. The base structure of EfficientNet is found using structural search and then scaled using the rules of compound scaling to obtain a series of networks with excellent performance. By using MBCConv from MobileNet V2 [16] as the backbone of the model, and squeeze excitation method from SENet to optimize the network structure, the performance of the model is further boosted. By scaling up baseline networks, the structure contains more parameters from the B0

[2]Figure from https://zhuanlan.zhihu.com/p/353235794
[3]Figure from https://arxiv.org/abs/1801.04381
[4]Figure from https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142

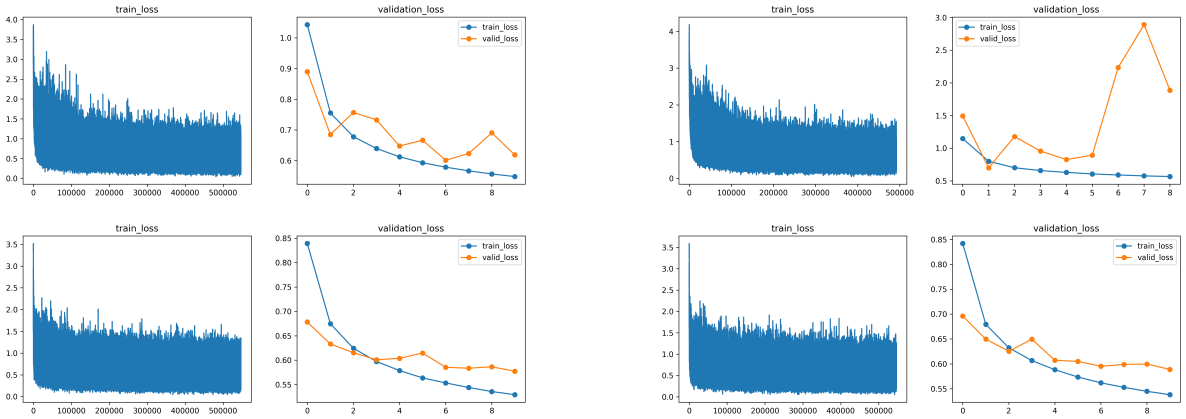


Figure 3: The training curves of ResNet-50 (top left), ResNet-101 (top right), pretrained ResNet-50 (bottom left), and pretrained ResNet-101 (bottom right)

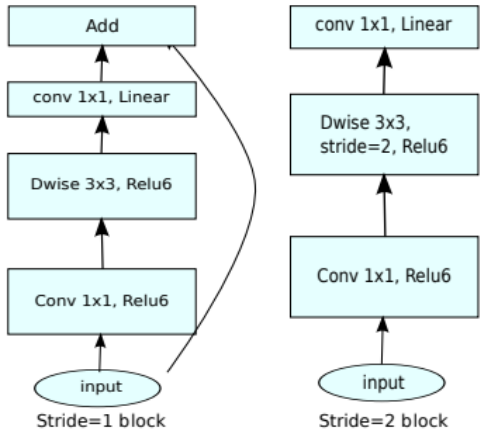


Figure 4: The architecture of MobileNet-V2

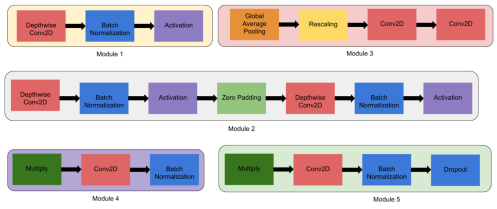


Figure 5: The modules of EfficientNet

version with about 11 million parameters, to the B7 version with about 65 million. We can refer to these results in Figure 7.

As our training samples are relatively scarce when compared with the parameters of ResNet, we also improve the models' performance by taking advantage of pretrained models. Pretraining has been a core idea in the recent development of transfer learning. It aims to leverage the huge amount of freely available datasets (such as ImageNet, a large-scale visual dataset with more than 14 million hand-annotated images) to acquire knowledge that is shared between similar tasks, and then utilize this knowledge to fine-tune on a small amount of training samples. In computer vision, this knowledge can be the feature representation of images. The models we use, provided by PyTorch, are pretrained on a 1000-category subset of ImageNet. To accommodate our 25-category training set, we substitute the last fully connected layer of ResNet (originally a mapping from $\mathbb{R}^{2048}$ to $\mathbb{R}^{1000}$) with a randomly initialized layer mapping from $\mathbb{R}^{2048}$ to $\mathbb{R}^{25}$.

For the purpose of comparison, all the networks are trained with the same hyperparameters: a learning rate of 3×10^{-4}, a batch size of 32, and a weight decay factor of 1×10^{-5}. The models are trained for 10 epochs on an RTX 3090, which took about three hours for MobileNet-V2, four hours for ResNet-50 and EfficientNet-B0, five and a half hours for ResNet-101, and fourteen hours for EfficientNet-B7.

3.3. Experimental Results

The training curves of ResNet-50, ResNet-101 and their respective pretrained versions are shown in figure 3, and their results are recorded in Table 1[5]. Among the models that are trained from scratch by us, ResNet-50 achieved a best performance of 50.43%. ResNet-101, on the other hand, have too many parameters and overfitted on the training set, as can be seen in Figure

[5]Due to limited time and computation resources, we did not conduct experiments with the non-pretrained versions of EfficientNet.

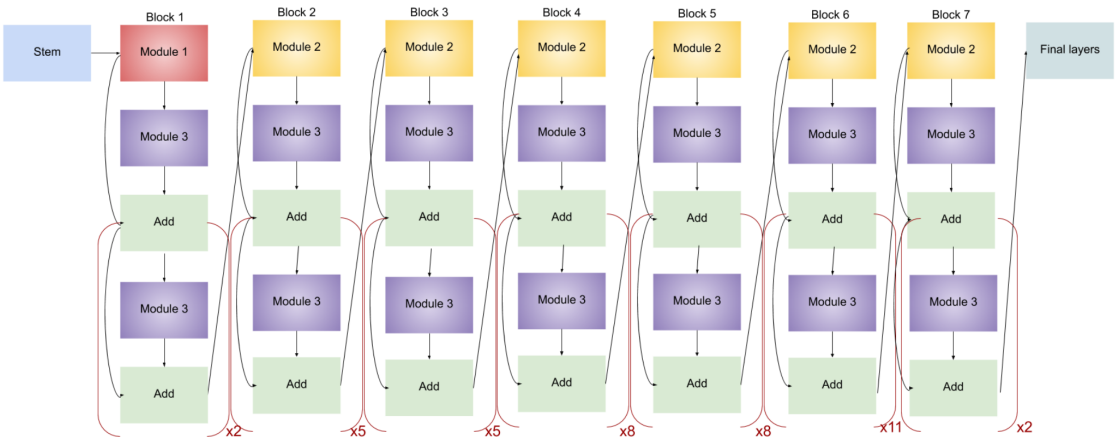


Figure 6: The structure of EfficientNet-B7

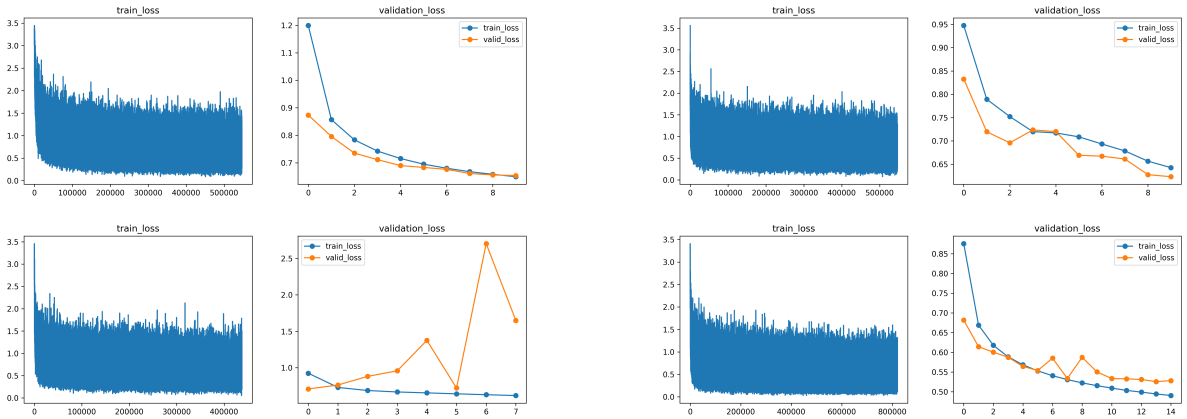


Figure 7: The training curves of MobileNet-V2 (top left), pretrained MobileNet-V2 (top right), pretrained EfficientNet-B0 (bottom left) and pretrained EfficientNet-B7 (bottom right)

3 where the validation loss starts to soar after several epoch of training while the training loss continues to drop steadily.

However, as mentioned above, the lack of training data can be mitigated by pretraning. As shown in Table 1, all models gained boost in performance when initialized with the pretrained weights, but ResNet-101 is the most prominent, whose test accuracy increased more than three percent, surpassing MobileNet and closing in on ResNet-50. And EfficientNet-B7, benefited by its shear network size, achieved the highest test score of 83.93%.

Table 1: The test accuracies of CNN models

Model	Acc	Acc (pretrained)
MobileNet-V2	80.36%	81.00%
ResNet-50	**81.43%**	82.13%
ResNet-101	78.86%	81.92%
EfficientNet-B0		79.42%
EfficientNet-B7		**83.93%**

3.4. Image Channel Redundancy

Sketch images generated from strokes are colorless, i.e. the images are black-and-white. Typical CNNs for image classification take RGB three-channel inputs, and in sketch classification task these three channels may be redundant. We investigate the redundancy by training and evaluating the models on the same set replacing the convolution kernel of the first layer with 1-channel-input kernel, and leaving other configurations unchanged. However, the results are practically the same as those in Table 1. This observation can be justified by the fact that the two redundant input channels only introduces several thousand extra parameters in

the first convolution layer, having almost no influence in face of the huge network architectures. Thus, we keep the three-channel input to follow the convention in image-related tasks.

Moreover, to further justify our conclusion, we visualize the first convolutional layer of ResNet-50 as in Figure 8. As can be seen from the figure where white pixels indicate the focus of the model, the weights barely differ across different channels. We also observe a redundant output channel in row 4, column 2 where all the pixels of the convolutional layer are black.

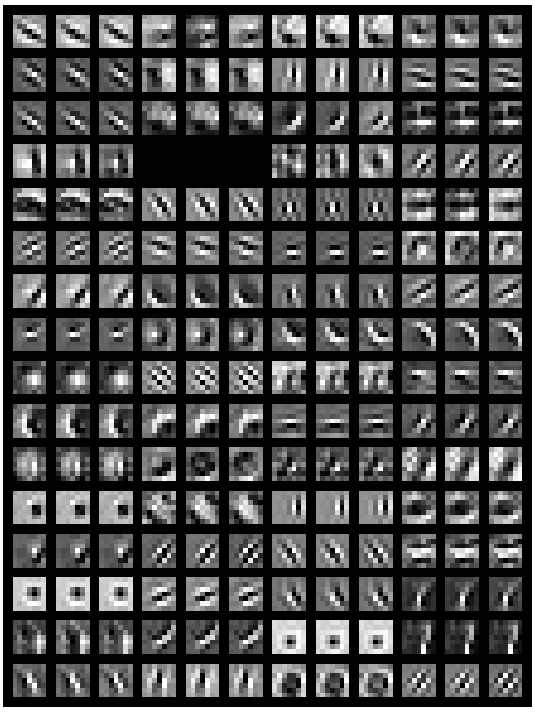


Figure 8: Weights of the 3-channel input first convolutional layer of ResNet-50

3.5. Robustness Analysis

To test the robustness of convolutional models, we flipped the test data, both horizontally and vertically, and ran inference again on them. The resulting test accuracy is recorded in Table 2.

Table 2: Robustness analysis of CNN models by flipping the test data. All models are pretrained.

Model	Flip Direction		
	horizontal	vertical	both
Resnet-50	81.16	26.46	26.49
MobileNet-V2	80.28	27.67	27.67
EfficientNet-B7	83.11	30.35	30.32

It can be observed that all the CNN models's test accuracy dropped dramatically when the input is flipped vertically, yet they are quite robust to horizontal flipping. The reason for such phenomena is two-fold. One is the inconsistency of drawers' focuses, for example focusing on the whole body or focusing on certain parts like head instead. This can be corroborated by roughly browsing the dataset. For instance, Figure 9 shows several possible patterns occurring in the cow sketches. The users of *Quick, Draw!* may focus on the head (column 1), the head and neck (column 2), whole body (column 3), or the texture (column 4) of the given animal. Therefore, without external information, the model has to try to guess what the focus is, and the prediction is based on all kinds of possible focuses.

Figure 9: Sample cow sketches with different focuses

And the other reason is the high abstraction of sketches, which makes them not vertically rotation-invariant even to human inspection. For example, Figure 10 shows a sketch of a dog and its flipped images. The predictions from trained ResNet-50 are shown in the right. When flipped horizontally, the model is still able to recognize the dog. However, when the image is flipped vertically, the ears of the dog are now recognized as the wings of an owl.

Figure 10: Effect of flip operation on a sample sketch (in the right of each column are the top-3 predictions of trained ResNet-50)

We also investigate the models' robustness from another perspective, by adding pixel-level Gaussian noise with 0 mean and 0.4 standard deviation to the test data as in Figure 11. It can be observed that noise of such scale does not visibly change the image's object. The testing results are shown in Table 3. While ResNet proved to be robust to noise, the performance of both MobileNet and EfficientNet dropped visibly. This is because the first convolution layer of ResNet uses 7×7 kernels, while the first layers of MobileNet and EfficientNet use 3×3 small kernels.

Filters with larger kernel sizes intuitively will smooth the features, making the models less sensitive to pixel-wise noise. Given a $k \times k$ convolutional kernel $\mathbf{K}$, we analyze the resulting feature output y of perception field $\mathbf{x}$ with noise ϵ, where the ϵs are independent and identically distributed Gaussian noise with mean 0 and variance σ^2. When the kernel parameters are normalized, larger k often implies smaller $\sum_{i=1}^{k}\sum_{j=1}^{k} K_{ij}^2$ from Equation 7, and the noise is more likely to be eliminated.

$$\begin{aligned} y &= \sum_{i=1}^{k}\sum_{j=1}^{k} K_{ij}(x_{ij} + \epsilon_{ij}) \\ &= G\left(y \middle| \sum_{i=1}^{k}\sum_{j=1}^{k} K_{ij}x_{ij}, \sigma^2 \sum_{i=1}^{k}\sum_{j=1}^{k} K_{ij}^2\right) \end{aligned} \tag{7}$$

More illustrative results can be seen in Figure 15, where we visualize the outputs of the three models with the same image input. The input image pair is that in Figure 11. Focusing on the effect of the noise, especially the pixels around the center and on the right of the image, we can see that the noise has little effect on the extracted features of ResNet in Figures 15(a) and 15(b). ResNet successfully denoises the data in the first filter layer while MobileNet and EfficientNet fail to eliminate the noise since Figures 15(d) and 15(f) are relatively noisy concerning the original outputs.

Table 3: Robustness analysis of CNN models by adding random noise. All models are pretrained.

	MobileNet-V2	ResNet-50	EfficientNet-B7
Acc	73.21	80.37	77.50

(a) Original Image (b) Noised Image

Figure 11: Example for adding noise to an image

To summarize, due to the abstraction of sketches, the models are robust to horizontal flipping and perform badly in vertical flipping due to semantic shift. Moreover, there is a trade-off in convolutional kernel choice between capturing locality and denoising, and ResNet is more robust to noise while EfficientNet is more accurate.

4. Classification with the Hybrid of CNN and RNN

4.1. Experimental Settings

Since sketches have distinct properties compared with natural images in that they are composed of only crude lines, in this section we leverage the stroke sequence representation of sketches and apply recurrent network to help with classification. As mentioned in Section 3.1, the strokes of QuickDraw are three-channel sequences with the first two ranging from about -30000 to 30000 indicating the pen's movement at each step, and the third being binary values indicating the pen's status. To prepare the data for RNN models, we simply perform Z-score normalization on the first two channels.

4.2. Methods

4.2.1 RNN-only Baseline

As an RNN baseline, we first test the performance of pure RNN on sketch recognition task. We first apply one-dimensional convolution on each channel of the input data to smooth the strokes and better capture tem-

poral relations of the points, and then feed the strokes into a two-layer LSTM with 256 hidden units. The second LSTM layer's output at each step is collected and averaged, the result of which is then used as the feature for classification task, as shown in Figure 12 [6].

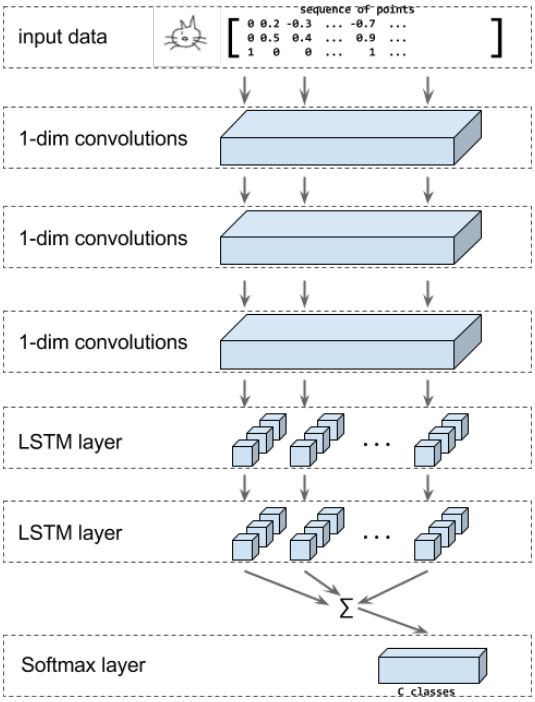


Figure 12: Network structure of baseline RNN

4.2.2 Integration of RNN into CNN

We then integrate RNN into our CNN models to help train a better classifier. A naïve idea is simply averaging the scores from RNN and CNN outputs, and we did not expect such idea to work well. Yet the experimental results are unexpectedly outperforming, reaching an accuracy of **84.49%** on the test set. Our explanation is that when a sketch is too abstract to recognize, CNN is uncertain and requires extra stroke sequence information extracted by RNN, though noised, as an auxiliary to determine the prediction.

However, as our RNN baseline alone only achieved a test accuracy of 57.25%, suggesting that the temporal information contained in strokes sequences may not be that helpful in sketch recognition, we design our network architectures carefully so as to make sure that RNN serves indeed only as an auxiliary and does not dominates over CNN. One choice of integrating RNN with CNN is the cascaded architecture proposed by [12], where a neural network based rasterizer transforms the features extracted by RNN into pixelated images, which are then sent into a CNN (Figure 13). However, our experiments show that in this way RNN brings too much unstability into the produced images, resulting in poor performance of the downstream CNN classification.

An alternative is the idea of dual-branched network architecture proposed in [24] (Figure 13), i.e. train a CNN model and an RNN model in parallel using the image input and stroke input respectively, and concatenate their extracted feature vector for the final classification. In this way, the model can utilize both the spatial structures of the sketch images and the temporal orders in which the lines are drawn, but also relies mainly on the CNN branch as the feature vector extracted by CNN is in a much higher dimension than that of RNN branch. We follow this framework, and choose pretrained EfficientNet-B7 for the CNN branch, with the same hyperparameters as described in Section 3.2. The RNN branch is trained with the same batch size as that of CNN, but with a learning rate of 3×10^{-3} and a weight decay factor of 1×10^{-4}. The features extracted by the two branches (2560-dimensional for CNN and 256-dimensional for RNN) are than concatenated and undergo a two-layer perceptron classifier.

Moreover, we extend the idea of pretraining by first training the CNN branch and RNN branch separately, and then with the two branches fixed learn a branch-level attention by fine-tuning the classifier. To save training time, we run the CNN and RNN on the training set and save the extracted features locally to avoid redundant forward passing through the two branches in each training step. Then we directly treat the intermediate features as input and train the classifier.

4.3. Results and Analysis

As mentioned before, our RNN baseline achieved an accuracy of 57.25% on the test set. This is not even close to the CNNs' performance, but still a satisfactory result considering that the strokes may be very difficult to recognize without accessing the spatial information, as the representation of strokes in the Cartesian coordinate of the same category may dramatically vary from each other even with a slight rotation. The naïve fusion of CNN and RNN's decision values yields a result of 84.49%, and when adjusted by our branch-level attention weights that score is pushed further up by half a percent, to **84.97%**. The results are summarized in Table. 4.

[6] Figure from https://github.com/tensorflow/docs/blob/master/site/en/r1/tutorials

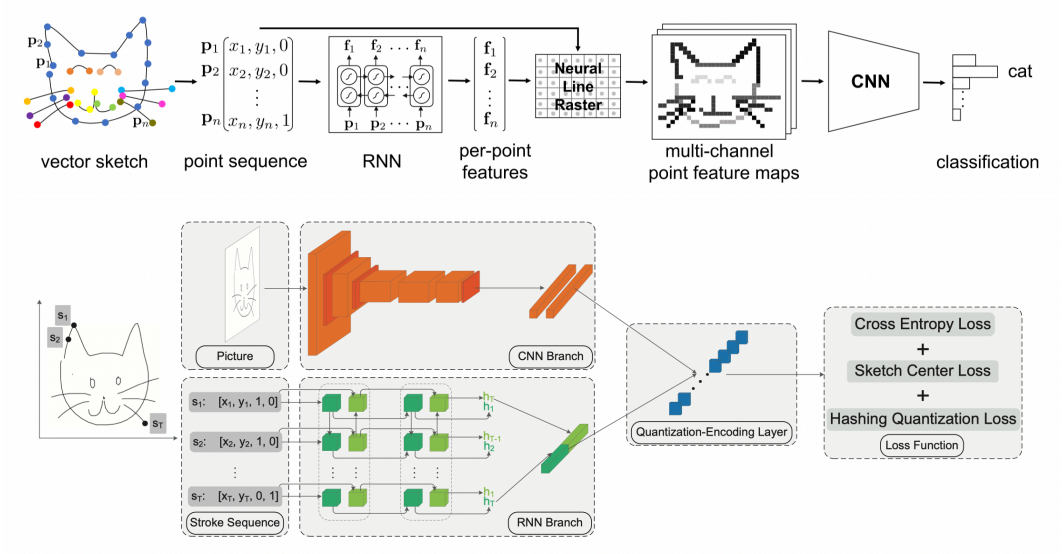


Figure 13: An illustration of cascaded RNN-CNN architecture (top) and dual-branch CNN-RNN architecture (bottom) for sketch recognition.

Table 4: Summary on test accuracy

	CNN	RNN	Naïve Fuze	Branch Attn
Acc	83.93%	57.25%	84.49%	84.97%

4.4. Ablation Studies

To understand the importance of each part of our RNN branch, we performed ablation studies on the LSTM model. As shown in Figure 12, we average the outputs of all the time steps from the second-layer LSTM before feeding them to the classifier. This operation turns out to be vital in training the RNN model. When feeding only the last-step output to the classifier, as is the common practice in many sequential tasks, the model fails to learn anything, as shown in Figure 14. We also find that first applying one-dimensional convolution to the inputs' channels before feeding them to LSTM helps increase the model's generalization ability, as can be seen in Figure 14, where the validation curve becomes jaggy after removing the convolution layers.

5. Limitations

Despite the model performance we achieved, there are two drawbacks of our work that requires further experiments to perfect if time and resource permit.

One is the average operation on the LSTM outputs. By this operation we assume that the outputs of all steps are equally weighted. However, the information extracted in different steps are not necessary equal in status intuitively. Therefore, more promising methods may be replacing the average operation with a self-attention layer, which consumes large GPU memory and computation time for such long sequences as in QuickDraw.

The other is the failure of training cascaded RNN-CNN models. We tried to reproduce the experiment of current SOTA model - Sketch-R2CNN - for sketch classification on the subset of QuickDraw with the same training configuration as used in the original paper, yet in the training process we did not see any trend of loss decreasing to lower than 3. The loss quickly converged to around 3.2, i.e. approximately $\ln 25$, suggesting that the model is randomly guessing the output. The underlying reason may be the insufficient amount of data and the large proportion of noise therein.

6. Conclusion

Sketch itself with high abstraction is not easily recognized, as can be seen in the game of *Draw Something*. The sparsity of sketch images makes sketch recognition even more challenging. In this project, we investigate the performance of several well-performing CNNs including ResNets, MobileNet, and EfficientNets on sketch recognition. Pretrained weights are used in initialization to boost the performance. Among them EfficientNet-B7 outperforms the others, reaching an accuracy of **83.93%** on the test set. To achieve better performance, we introduce an LSTM trained on the stroke sequences with a relatively low performance of 57%. However, by introducing a 2-layer perceptron as

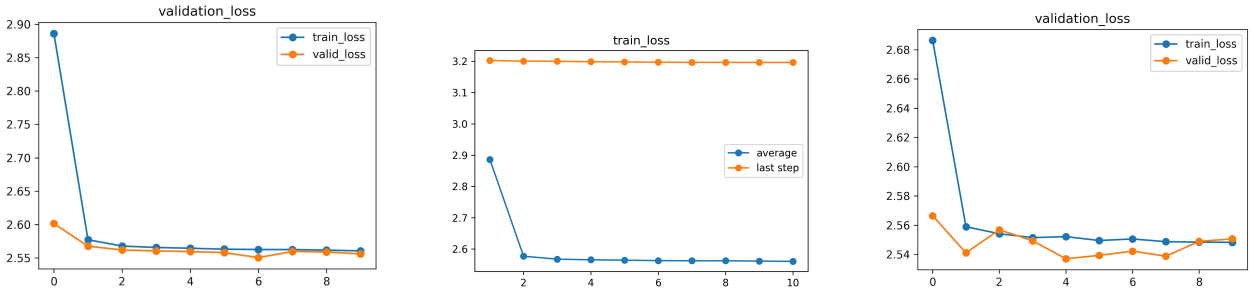


Figure 14: Ablation studies on LSTM: the training and validation curves of the RNN model adopted by us (left) compared with (1) taking last-step output (middle), and (2) without one-dimensional convolution layers (right).

dual-branch attention module, the LSTM helped EfficientNet to push the test accuracy up to **84.97%**. We also conduct robustness analysis on the CNN models and ablation studies on the RNN model, and analyze their respective strengths and weaknesses.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[2] A. D. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[3] D. Ha and D. Eck. A neural representation of sketch drawings. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[6] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[7] 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.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] J. Li, J. Wang, Q. Tian, W. Gao, and S. Zhang. Global-local temporal representations for video person re-identification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3958–3967, 2019.

[11] X. Li, W. Wang, X. Hu, and J. Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 510–519, 2019.

[12] Z. Y. e. a. Li L, Zou C. Sketch-r2cnn: An rnn-rasterization-cnn architecture for vector sketch recognition. *IEEE transactions on visualization and computer graphics*, 3745-3754., 2020.

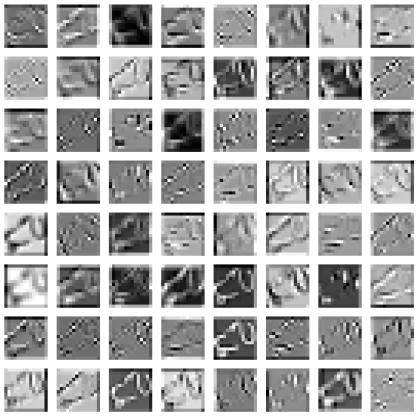
[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[14] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

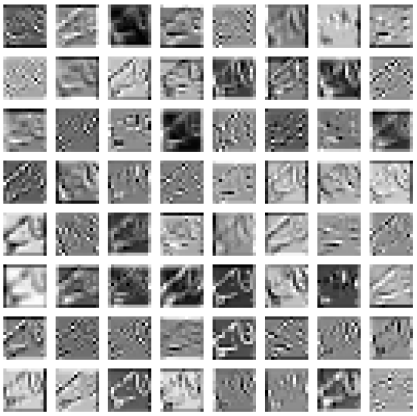
[15] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. pages 4510–4520, 06 2018.

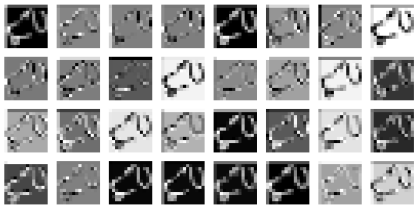
[17] R. K. Sarvadevabhatla and J. Kundu. Enabling my robot to play pictionary: Recurrent neural networks for sketch recognition. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 247–251, 2016.



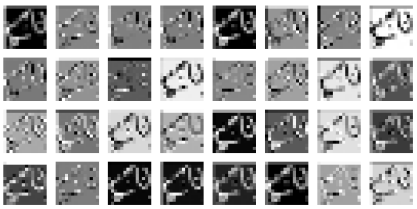
(a) Original 1st Layer Feature, ResNet



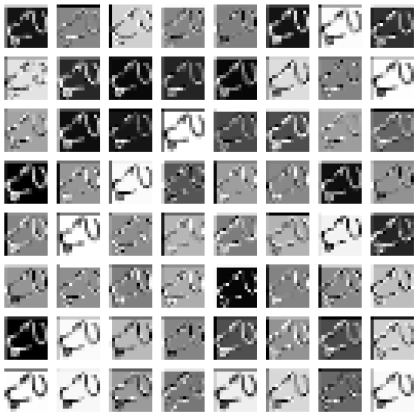
(b) Noised 1st Layer Feature, ResNet



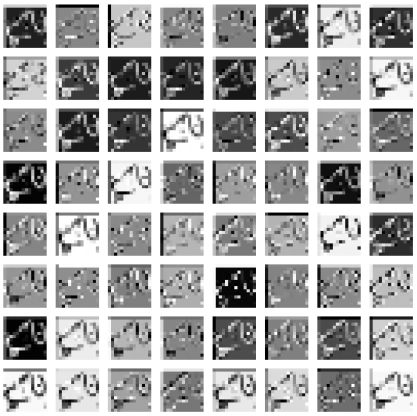
(c) Original 1st Layer Feature, MobileNet



(d) Noised 1st Layer Feature, MobileNet



(e) Original 1st Layer Feature, EfficientNet



(f) Noised 1st Layer Feature, EfficientNet

Figure 15: Features Extracted by the First Layer Filters of ResNet, MobileNet and EfficientNet

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[19] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, abs/1905.11946, 2019.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[21] F. Wang, S. Lin, H. Wu, H. Li, R. Wang, X. Luo, and X. He. Spfusionnet: Sketch segmentation using multi-modal data fusion. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1654–1659. IEEE, 2019.

[22] P. Xu, T. M. Hospedales, Q. Yin, Y.-Z. Song, T. Xiang, and L. Wang. Deep learning for free-hand sketch: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[23] P. Xu, C. K. Joshi, and X. Bresson. Multigraph transformer for free-hand sketch recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[24] Y. T. e. a. Xu P, Huang Y. Sketchmate: Deep hashing for million-scale human sketch retrieval. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8090-8098., 2018.

[25] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales. Sketch-a-net that beats humans. *arXiv preprint arXiv:1501.07873*, 2015.

[26] X. Zhang, Y. Huang, Q. Zou, Y. Pei, R. Zhang, and S. Wang. A hybrid convolutional neural network for sketch recognition. *Pattern Recognition Letters*, 130:73–82, 2020.