A Novel Sketch Recognition Model based on Convolutional Neural Networks

Abdullah Talha Kabakus

Department of Computer Engineering, Faculty of Engineering, Duzce University, Duzce, Turkey

talhakabakus@duzce.edu.tr

Abstract— **Deep neural networks have been widely used for visual recognition tasks based on real images as they have proven their efficiency. Unlike real images, sketches exhibit a high level of abstraction as they lack the rich features that the real images contain such as various colors, backgrounds, and environmental detail. Despite all of these shortages and being drawn with just a few strokes, they are still meaningful enough to encompass an appropriate level of meaning. The efficiency of deep neural networks on sketch recognition has been relatively less studied compared to the visual recognition of real images. To experiment with the efficiency of deep neural networks on sketch recognition, a novel sketch recognition model based on Convolutional Neural Networks is proposed in this study. The proposed model consisted of 21 layers and was tuned in an automated manner to find out the best-optimized model. In order to reveal the proposed model's efficiency in terms of predicting the classes of the given sketches, the model was evaluated on a gold standard sketch dataset, namely, *Quick, Draw!*. According to the experimental result, the proposed model's accuracy was calculated as high as 89.53% which outperformed the related work on the same dataset. The key findings that were obtained during the conducted experiments were discussed to shed light on future studies.**

Keywords—sketch recognition, image recognition, convolutional neural network, deep neural network, deep learning

I. INTRODUCTION

Deep neural networks have proven their efficiency for several tasks such as facial expression recognition [1], medical diagnosis (e.g., diagnosis of breast cancer) [2], and face detection on the real images which are pixel perfect depictions and contain rich features such as various colors, background, and environmental detail. When it comes to sketches, they exhibit a high level of abstraction despite that they are surprisingly meaningful as with just a few strokes, they are able to encompass an appropriate level of meaning [3]. Humans learn abstractions that store important pieces of information. [4] In other words, human brains are only able to concentrate consciously on a few things at once [5]. As a natural result of this ability, humans are able to detect objects in various conditions (e.g., different viewpoints, different colors, or shapes of the same object). In this direction, in the study [6], a person was asked to sketch a one-dollar bill. As the result is presented in Fig. 1 with the drawing the person subsequently made while a one-dollar bill was present, the person only remembered the most important visual parts of it. So, the quality of the abstractions learned is believed to be a critical factor in the flexibility and adaptability of humans [4].

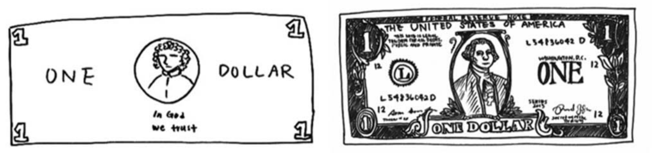

Fig. 1. On the left, a person's sketch of a dollar bill is presented. On the right, the subsequent sketch of the same person while a dollar bill was present is presented [6].

In addition to these unique characteristics of sketches, the prevalence of touchscreen devices (e.g. tablets, smartphones) has also increased the ability to draw and use sketches in many applications such as web icons in websites/web applications, illustrations, and animated movies. Sketch-based image recognition is a more challenging task compared to image recognition based on real images due to the absence of rich features that real images do contain. To this end, a novel deep neural network model based on a convolutional neural network (CNN) is proposed within this paper. The proposed model was evaluated on a gold standard sketch dataset to reveal its efficiency. The rest of this paper is structured as follows: Section 2 presents the related work. Section 3 describes the material and method for the proposed novel model. Section 4 presents the experimental result and discussion. Finally, Section 5 concludes the paper with future direction.

II. RELATED WORK

Some researchers [7]–[9] proposed sketch recognition models, which are largely based on hand-crafted features, and utilized the SVM (Support Vector Machine) as the classifier of their models. *Yue et al.* [10] proposed an approach, namely, *Sketch-a-Net (SN1.0)*, based on the utilization of the discriminative power of deep neural networks. A subsequent approach, namely, *Sketch-a-Net (SN2.0)*, investigated stroke-level temporal information by applying data augmentation strategies [11]. *Sketch-R2CNN* [12] takes as input only a vector sketch with grouped sequences of points and uses an RNN (Recurrent Neural Network) for stroke attention estimation in the vector space and a CNN for 2D feature extraction in the pixel space respectively. *SketchMate* [3] jointly investigates static sketch visual characteristics and dynamic temporal sketching information in a single model. *SketchMate* consists of a two-branch CNN-RNN deep neural network where the CNN encoder is responsible for extracting the features from the given images, the RNN encoder takes in the vectors of stroke sequences and outputs its final time-step state. *Wang et al.* [13] proposed an approach based on two Siamese CNNs, one for the views, and the one for the sketches. *Sarvadevabhatla et al.* [14] proposed an RNN architecture for sketch object recognition which utilized the long-term sequential and structural regularities in the strokes of sketches. *Xu et al.* [15] proposed a new approach based on the idea of representation of sketches as multiple sparsely connected graphs. Then, they design a novel GNN (Graph Neural Network) in order to learn representations of sketches from multiple graphs. *Prabhu et al.* [16] proposed an approach for sketch recognition based on the distribution-aware binarization of neural networks. *He et al.* [17] proposed a sketch recognition model, namely, *Deep Visual-Sequential Fusion model (DVSF)*, that utilized both R-FC (Residual Fully-Connected) and R-LSTM (Residual Long Short-Term Memory) layers. While R-FC layers represented the visual networks and were responsible for learning the stroke patterns, R-LSTM layers represented the sequential networks and were responsible for learning the patterns of stroke order. Finally, the visual and sequential representations of sketches were integrated with a fusion layer.

III. Material and Method

In this section, both the proposed novel deep neural network model which is based on a Convolutional Neural Networks and the dataset that this model was evaluated on are described.

A. Dataset Construction

The dataset which was utilized to evaluate the proposed model was the *Quick, Draw!* dataset [18], which is provided by *Google Creative Lab* as a part of their experiment[1] of recognizing sketches. This dataset consists of 345 classes and was collected by asking more than 15 million volunteers from all around the world. Each volunteer was given 20 seconds to quickly sketch a given class. According to *Google*, this dataset is the world's largest sketch dataset with having 50 million drawings. Hence, the proposed study utilized the *Quick, Draw!* dataset instead of the other well-known datasets such as the *TU-Berlin* [7], and the *Sketchy* [19] databases, that consist of $20k$, and $75k$ sketches, respectively. Another reason for preferring the *Quick, Draw!* dataset instead of the other databases was that the *Quick, Draw!* dataset is highly noisy due to (i) the users had only 20 seconds to sketch, and (ii) no specific post-processing was performed on it [3]. Within this study, a sub dataset of the *Quick, Draw!* dataset was constructed in the same way *Lamb et al.* [4] did as follows: Each class in the widely-used *CIFAR-10* dataset was mapped with the classes of the *Quick, Draw!* dataset. The unmatching classes were discarded, and when the relevant classes were available, they were included as well. For example, the "$automobile$" class in the *CIFAR-10* dataset was mapped with the "car", and "$police\ car$" classes. Since there is no appropriate mapping for the "$deer$" class of the *CIFAR-10* dataset in the *Quick, Draw!* dataset, this class was discarded. Consequently, the constructed dataset consisted of nine classes, namely, "$airplane$", "$automobile$", "$bird$", "cat", "dog", "$frog$", "$horse$", "$ship$", and "$truck$". For each *Quick, Draw!* class, 10,000 drawings were sampled from the *Quick, Draw!* dataset. Therefore, the constructed dataset contains 210,000 drawings. The class mappings between the constructed dataset and the *Quick, Draw!* dataset are listed in Table 1. Some sample images from the constructed dataset for the classes "$airplane$", "$automobile$", "$bird$", "cat", "dog", "$horse$" and "$ship$", respectively, are presented in Fig. 2.

TABLE I. The Class Mapping Between the Constructed Dataset and the *Quick, Draw!* Dataset

Constructed Dataset Class	***Quick, Draw!* Class**
airplane	*airplane*
automobile	*car*
	police car
bird	*bird*
	duck
	flamingo
	owl
	parrot
	penguin
	swan
cat	*cat*
	lion
	tiger
dog	*dog*
frog	*frog*
horse	*horse*
ship	*cruise ship*
	sailboat
	speedboat
truck	*truck*
	firetruck

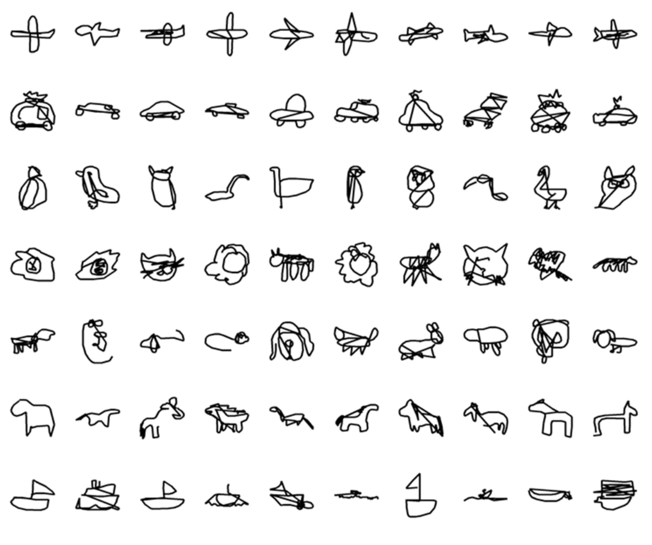
Fig. 2. Some sample images from the constructed dataset for the classes "$airplane$", "$automobile$", "$bird$", "cat", "dog", "$horse$" and "$ship$", respectively.

[1] The experiment is available online at https://quickdraw.withgoogle.com.

B. The Proposed Model

The proposed novel model is based on a convolutional neural network and responsible for both (i) learning the stroke patterns of sketches, and (ii) predicting the classes of the given sketches. For the implementation of the proposed model, a widely-used, open-source high-level deep neural networks library written in Python, namely, *Keras* [20], was utilized. *Keras* is capable of running on the top of various backends such as *TensorFlow* [21], *Theano*, and *Microsoft Cognitive Toolkit (CNTK)*. Within this study, *TensorFlow*, which is an open-source machine learning platform developed by *Google*, was opted as the backend because of being the recommended one by the developer of *Keras* [22]. The proposed model was trained on the *Google Colab* [23] platform since this platform provides powerful GPUs, such as the *Nvidia Tesla K80*, which is a key necessity for the training of deep neural networks because of demanding too much computational power. The other advantages of using the *Google Colab* are: (1) The seamless integration with the *Google Drive*, which provides permanent storage for the datasets, and (2) the platform already provides various highly popular Python libraries related to data science including but not limited to *TensorFlow*, *Keras*, *scikit-learn*, and *matplotlib*. In addition to this, additional packages can be installed through the *pip*, which is the package installer for Python.

The *Quick, Draw!* dataset provides the sketches in a 28x28 grayscale bitmap in *NumPy* format *(.npy)*, which is a format of a highly popular Python library, namely, *NumPy* [24], that provides both multi-dimensional arrays and matrices, and a large collection of high-level mathematical functions to operate on these data structures. Therefore, the dimension of the input was $28x28x1$. $10{,}000$ drawings for each class in the constructed dataset were extracted from these provided *NumPy* files and stored as image files. The proposed model starts with a convolution layer (denoted with $Conv2D$) which accepts the input and performs the convolution operations on. After the first sequential convolutional layers, a $MaxPooling$ layer, which is responsible for progressively reducing the spatial size of the representation that helps to reduce the number of parameters and computation in the network [25], was employed. $Dropout$ [26] is a widely-used technique that was proposed to prevent the 'overfitting' problem, which is one of the biggest challenges of deep neural networks [27], [28]. In order to prevent overfitting, several $Dropout$ layers were employed in various positions. In addition to this, $MaxPooling$ layers also help to control overfitting [25]. $Batch\ Normalization$ (denoted with $Batch\ Norm.$) layers were employed to normalize the activations of the previous layer in each batch. After the convolutions were performed, the dimension transformation was carried out by the $Flatten$ layer, which prepares the data to be passed to the $Dense$ layers. Then, the two $Dense$ layers, which are deeply connected neural network components, were employed. The last layer, namely, the *Softmax* layer, was responsible for the classification of the given sketch as one of the pre-defined nine classes. The *Rectified Linear Units (ReLU)* was employed as the activation function for all the activations except for the last layer since it provided the best accuracy during the hyper-parameter optimization. The detailed information about each layer in the proposed model is listed in Table 2.

TABLE II. THE DETAILED INFORMATION ABOUT EACH LAYER IN THE PROPOSED MODEL

#	Layer Name	Layer Parameters	Number of Parameters	Output Shape
0	$Input$	–	–	$28x28x1$
1	$Conv2D$	filters=16, kernel size=(3,3), activation=*ReLU*	320	$28x28x16$
2	$Conv2D$	filters=32, kernel size=(3,3), strides=(1,1), activation=*ReLU*	9,248	$26x26x32$
3	$Batch\ Norm.$	momentum=0.99	128	$26x26x32$
4	$MaxPooling$	pool size=(2,2), strides=(2,2)	–	$13x13x32$
5	$Dropout$	rate=0.4	–	$13x13x32$
6	$Conv2D$	filters=64, kernel size=(3,3), strides=(1,1), activation=*ReLU*	18,496	$11x11x63$
7	$Conv2D$	filters=128, kernel size=(3,3), strides=(1,1), activation=*ReLU*	36,928	$9x9x128$
8	$Batch\ Norm.$	momentum=0.99	256	$9x9x128$
9	$MaxPooling$	pool size=(2,2), strides=(2,2)	–	$4x4x128$
10	$Dropout$	rate=0.4	–	$4x4x128$
11	$Flatten$	–	–	2048
12	$Dense$	units=512, activation=*ReLU*	524,800	512
13	$Batch\ Norm.$	momentum=0.99	2,048	512
14	$Dropout$	rate=0.4	–	512
15	$Dense$	units=512, activation=*ReLU*	262,656	512
16	$Batch\ Norm.$	momentum=0.99	2,048	512
17	$Dropout$	rate=0.4	–	512
18	$Dense$	units=32, activation=*ReLU*	16,416	32
19	$Batch\ Norm.$	momentum=0.99	128	32
20	$Dropout$	rate=0.4	–	32
21	$Dense$	activation=*Softmax*	297	9

The *Adaptive Moment Estimation (Adam)* algorithm [29], which is an extension of the *Stochastic Gradient Descent (SGD)*, was employed as the optimization algorithm of the model which computes adaptive learning rates for each network parameter from estimates of first and second moments of the gradient [2]. A loss function in deep neural networks is responsible for estimating the loss of the model in order to reduce the loss on the next iteration. The *Categorical Cross-Entropy* (*a.k.a. Softmax Loss*) was employed as the loss function of the proposed model since it provided the best accuracy during the hyper-parameter optimization. The accuracy was selected as the evaluation metric of the model.

Hyper-parameters are the parameters of a deep neural network model that affect the learning process and are set empirically [22], [30]. To this end, an open-source Python library, which acts as a wrapper for hyperparameter optimization specifically designed for *Keras*, namely, *Hyperas* [31], was employed. For the proposed model, a wide range of values have experimented for the hyper-parameter optimization as they are listed in Table 2, and the ones that provided the best performance in terms of accuracy were highlighted in bold which should shed light on future studies on sketch recognition.

TABLE III. THE EXPERIMENTED VALUES OF HYPER-PARAMETERS DURING THE HYPER-PARAMETER OPTIMIZATION

Hyper-parameter	**Experimented Values**
Optimization algorithm	***Adam***, *SGD*, *RMSprop*
Loss function	***Categorical Cross-Entropy***, *Mean Absolute Error*, *Mean Squared Error*
Learning rate	e^{-2}, $\boldsymbol{e^{-3}}$, e^{-4}, e^{-5}
Dropout rate	0.2, 0.3, **0.4**, 0.5, 0.6, 0.7, 0.9
Activation function	***ReLU***, tanh, sigmoid
Kernel size	**3**, 5

IV. EXPERIMENTAL RESULT AND DISCUSSION

The proposed model's training was started with the two callbacks provided by *Keras*: (1) *EarlyStopping*, which stops the training when the monitored criterion is not getting better during the pre-defined number of epochs (*a.k.a. patience*), and (2) *ModelCheckPoint*, which saves the model weights into the local storage to let it be used later. During the training, validation loss was monitored, and the patience of the *EarlyStopping* was set to 10 epochs which means the training was stopped when the validation loss had not been decreased for 10 epochs. 20% of the whole dataset was utilized as the test set, and the remaining subset was split as follows: 20% of it was utilized as the validation set, the remaining 80% was utilized as the training set. The number of images for training, validation, and test sets are given in Table 4.

TABLE IV. THE NUMBER OF IMAGES FOR TRAINING, VALIDATION, AND TEST SETS

Set	**Number of images**
Training	134,400
Validation	33,600
Test	42,000

The training was stopped after 32 epochs since the validation loss had not been decreased for 10 epochs. After the training process was completed, the model was evaluated on the test set. The accuracy of the test set was calculated as high as 89.53%. The accuracies achieved on the training and validation sets were plotted in Fig. 3.

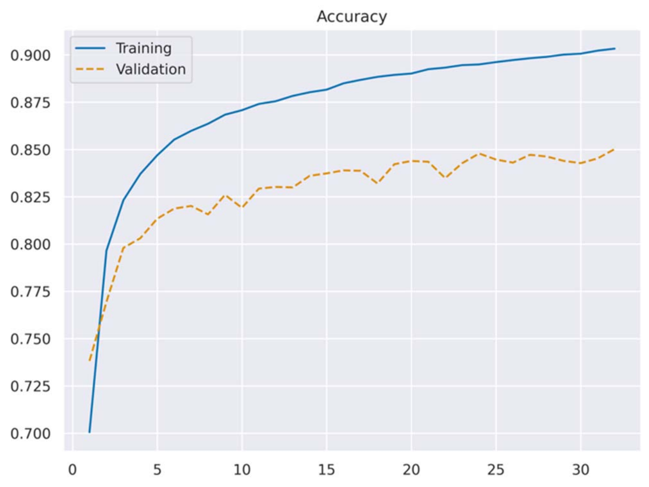

Fig. 3. The plot of the accuracies achieved on the training and validation sets.

The confusion matrix is the de facto standard of measuring the efficiency of classifiers in terms of the proposed system's ability to classify given samples to the classes they actually belong to [32]–[34]. Therefore, the confusion matrix of the test result was calculated. Based on the confusion matrix, the accuracy for each class with its most misclassified class is listed in Table 5. As the plot of the confusion matrix of the test result is presented in Fig. 4, the following conclusions were drawn:

- The proposed model's most accurately recognized class was found as the "*ship*" as 96.4% of the test images were correctly recognized. Following this, the second most accurately recognized class was found as the "*bird*" as 93.2% of the test images were correctly recognized.

- The most mismatched case occurred between the classes "*bird*" and "*frog*" as the 13.5% of the "*bird*" sketches were recognized as the "*frog*" sketches. Following this, the second most mismatches case occurred between the classes "*dog*" and "*cat*" as the 11.2% of the "*dog*" sketches were recognized as the "*cat*" sketches. This situation was also discussed by *Lamb et al.* [4] as follows: Many of the sketches from these classes focus on the face of the animal, and the only major difference between these animals is the shape of the ears, with *dogs* having rounded ears and *cats* having pointed ears. This highly salient feature may not be picked up by a deep neural network.

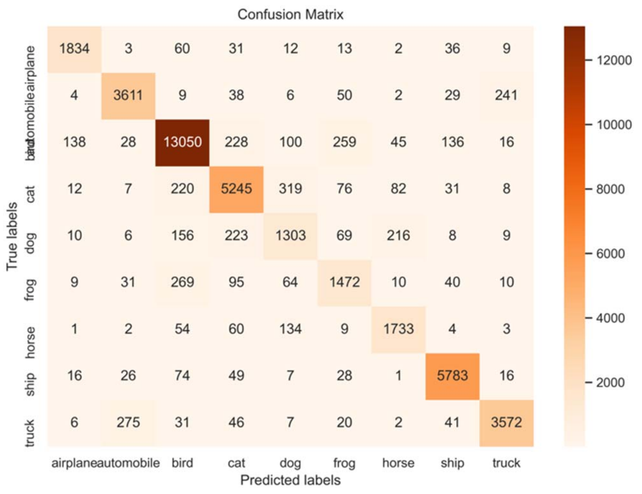

Fig. 4. The confusion matrix of the test result.

TABLE V. THE ACCURACY OF EACH CLASS WITH ITS MOST MISCLASSIFIED CLASS ON THE TEST SET

Class	Accuracy (%)	Most misclassified class with the misclassification ratio (%)
airplane	91.7	*bird* - 3
automobile	90.3	*truck* - 6
bird	93.2	*frog* - 1.9
cat	87.4	*dog* - 5.3
dog	65.2	*cat* - 11.2
frog	73.6	*bird* - 13.5
horse	86.7	*dog* - 6.7
ship	96.4	*bird* - 1.2
truck	89.3	*automobile* - 6.9

There is only one study whose dataset is the same as the proposed study which is the *SketchTransfer* [4]. Therefore, we can only compare the performance of the proposed model with it. The proposed model outperformed the related work on the same dataset by achieving an accuracy of 89.53% as the performance comparison in terms of achieved accuracy is given in Table 6.

TABLE VI. THE PERFORMANCE COMPARISON OF THE RELATED WORK ON THE SAME DATASET

Related Work	Accuracy (%)
SketchTransfer [4]	87.25
The proposed model	**89.53**

V. CONCLUSION

Deep neural networks have been widely used in visual recognition tasks and they have proven their efficiency in real images. Unlike real images, sketches exhibit a high level of abstraction and lack the rich features that the real images contain such as various colors, backgrounds, and environmental detail. Despite all of these shortages, sketches are still meaningful enough to encompass an appropriate level of meaning. However, the efficiency of deep neural networks on sketch recognition has been less experimented compared to the real images. To this end, a novel sketch recognition model based on Convolutional Neural Networks is proposed in this paper. The proposed model consisted of 21 layers and was tuned to find out the best-optimized model. The optimized model was evaluated on a gold standard dataset, namely, *Quick, Draw!*. According to the experimental result, the accuracy of the proposed model was calculated as high as 89.53% which is higher than the related work and proves the efficiency of Convolutional Neural Networks on sketch recognition.

As future work, the model may be enhanced by considering the patterns of stroke order. Also, the constructed dataset may be extended by including both more classes and more samples.

REFERENCES

[1] A. Majumder, L. Behera, and V. K. Subramanian, "Automatic Facial Expression Recognition System Using Deep Network-Based Data Fusion," IEEE Trans. Cybern., vol. 48, no. 1, pp. 103–114, 2018, doi: 10.1109/TCYB.2016.2625419.

[2] N. Brancati, G. De Pietro, M. Frucci, and D. Riccio, "A Deep Learning Approach for Breast Invasive Ductal Carcinoma Detection and Lymphoma Multi-Classification in Histological Images," IEEE Access, vol. 7, pp. 44709–44720, 2019, doi: 10.1109/ACCESS.2019.2908724.

[3] P. Xu et al., "SketchMate: Deep Hashing for Million-Scale Human Sketch Retrieval," in Proceedings of the 2018 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2018), 2018, pp. 8090–8098, doi: 10.1109/CVPR.2018.00844.

[4] A. Lamb, S. Ozair, V. Verma, and D. Ha, "SketchTransfer: A Challenging New Task for Exploring Detail-Invariance and the Abstractions Learned by Deep Networks," in Proceedings of the 2020 Winter Conference on Applications of Computer Vision (WACV '20), 2020, pp. 1–10.

[5] E. M. Gray and D. O. Tall, "Abstraction as a Natural Process of Mental Compression," Math. Educ. Res. J., vol. 19, no. 2, pp. 23–40, 2007, doi: 10.1007/BF03217454.

[6] R. Epstein, "The empty brain," Aeon, 2016. https://aeon.co/essays/your-brain-does-not-process-information-and-it-is-not-a-computer (accessed Jun. 06, 2020).

[7] M. Eitz, J. Hays, and M. Alexa, "How Do Humans Sketch Objects?," ACM Trans. Graph., vol. 31, no. 4, pp. 1–10, 2012, doi: 10.1145/2185520.2185540.

[8] Y. Li, T. M. Hospedales, Y. Z. Song, and S. Gong, "Free-hand sketch recognition by multi-kernel feature learning," Comput. Vis. Image Underst., vol. 137, pp. 1–11, 2015, doi: 10.1016/j.cviu.2015.02.003.

[9] R. G. Schneider and T. Tuytelaarsy, "Sketch classification and classification-driven analysis using Fisher vectors," in ACM Transactions on Graphics, 2014, doi: 10.1145/2661229.2661231.

[10] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales, "Sketch-a-Net that Beats Humans," in Proceedings of the 26th British Machine Vision Conference (BMVC 2015), 2015, doi: 10.5244/c.29.7.

[11] 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," Int. J. Comput. Vis., vol. 122, pp. 411–425, 2017, doi: 10.1007/s11263-016-0932-3.

[12] L. Li, C. Zou, Y. Zheng, Q. Su, H. Fu, and C.-L. Tai, "Sketch-R2CNN: An Attentive Network for Vector Sketch Recognition," arXiv Prepr., vol. 1811.08170, pp. 1–10, 2018.

[13] F. Wang, L. Kang, and Y. Li, "Sketch-based 3D Shape Retrieval using Convolutional Neural Networks," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2015), 2015, pp. 1875–1883, doi: 10.1109/CVPR.2015.7298797.

[14] R. K. Sarvadevabhatla, J. Kundu, and V. BabuR, "Enabling My Robot To Play Pictionary: Recurrent Neural Networks For Sketch Recognition," in Proceedings of the 24th ACM International Conference on Multimedia (MM '16), 2016, doi: 10.1145/2964284.2967220.

[15] P. Xu, C. K. Joshi, and X. Bresson, "Multi-Graph Transformer for Free-Hand Sketch Recognition," arXiv Prepr., vol. 1912.11258, pp. 1–14, 2020.

[16] A. Prabhu, V. Batchu, S. A. Munagala, R. Gajawada, and A. Namboodiri, "Distribution-Aware Binarization of Neural Networks for Sketch Recognition," in Proceedings of the 2018 IEEE Winter

Conference on Applications of Computer Vision (WACV 2018), 2018, pp. 1–12, doi: 10.1109/WACV.2018.00096.

[17] J. Y. He, X. Wu, Y. G. Jiang, B. Zhao, and Q. Peng, "Sketch Recognition with Deep Visual-Sequential Fusion Model," in Proceedings of the 25th ACM International Conference on Multimedia (MM '17), 2017, pp. 448–456, doi: 10.1145/3123266.3123321.

[18] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg, "The Quick, Draw!-AI Experiment," Google AI Experiments, 2017. https://experiments.withgoogle.com/quick-draw (accessed Jun. 06, 2020).

[19] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, "The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies," ACM Trans. Graph., vol. 35, no. 4, pp. 1–12, 2016, doi: 10.1145/2897824.2925954.

[20] F. Chollet, "Keras," 2020. https://keras.io (accessed Jun. 08, 2020).

[21] "TensorFlow," Google, 2020. https://www.tensorflow.org (accessed Jun. 08, 2020).

[22] F. Chollet, Deep Learning with Python. Manning Publications, 2017.

[23] "Google Colab," Google, 2020. https://colab.research.google.com (accessed Jun. 10, 2020).

[24] T. E. Oliphant, A Guide to NumPy. Trelgol Publishing, 2006.

[25] "CS231n Convolutional Neural Networks for Visual Recognition," Stanford University, 2020. https://cs231n.github.io/convolutional-networks (accessed Jun. 08, 2020).

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.

[27] M. Amin, B. Shah, A. Sharif, T. Ali, K. l. L. Kim, and S. Anwar, "Android malware detection through generative adversarial networks," Trans. Emerg. Telecommun. Technol., vol. e3675, pp. 1–29, 2019, doi: 10.1002/ett.3675.

[28] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," J. Ambient Intell. Humaniz. Comput., vol. 10, no. 8, pp. 3035–3043, 2019, doi: 10.1007/s12652-018-0803-6.

[29] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in 3rd International Conference on Learning Representations (ICLR 2015), 2015, pp. 1–15.

[30] X. Zhang, X. Chen, L. Yao, C. Ge, and M. Dong, "Deep Neural Network Hyperparameter Optimization with Orthogonal Array Tuning," in International Conference on Neural Information Processing (ICONIP 2019), 2019, pp. 287–295, doi: 10.1007/978-3-030-36808-1_31.

[31] M. Pumperla, "Hyperas by maxpumperla," 2020. http://maxpumperla.com/hyperas/ (accessed Jun. 08, 2020).

[32] T. C. W. Landgrebe, P. Paclik, R. P. W. Duin, and A. P. Bradley, "Precision-Recall Operating Characteristic (P-ROC) curves in imprecise environments," in Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), 2006, pp. 1–5, doi: 10.1109/ICPR.2006.941.

[33] I. Düntsch and G. Gediga, "Confusion Matrices and Rough Set Data Analysis," J. Phys. Conf. Ser., vol. 1229, pp. 1–6, 2019, doi: 10.1088/1742-6596/1229/1/012055.

[34] K. Sikka, A. Dhall, and M. Bartlett, "Exemplar Hidden Markov Models for Classification of Facial Expressions in Videos," in 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, pp. 18–25, doi: 10.1109/CVPRW.2015.7301350.