Supplementary Materials for "Convolutional Neural Network Based Ensemble Approach for Homoglyph Recognition"

Md. Taksir Hasan Majumder, Md. Mahabur Rahman, Anindya Iqbal, M. Sohel Rahman

Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology,
West Palashi, Dhaka - 1000, Bangladesh

---

1. Performance Metrics

Definition 1. ***Positive (P)****. The number of real positive cases/samples.*

Definition 2. ***Negative (N)*** *The number of real negative cases/samples.*

Definition 3. ***True Positive (TP)*** *Number of correctly predicted positive values (i.e., both the values of the actual class and the predicted class are yes.)*

Definition 4. ***False Positive (FP)*** *Number of mistakes in predicted positive values (i.e., the value of the actual class is no, however the value of the predicted class is yes.)*

Definition 5. ***True Positive (TN)*** *Number of correctly predicted negative values (i.e., both the values of the actual class and the predicted class are no.)*

Email addresses: 0905002.mthm@ugrad.cse.buet.ac.bd (Md. Taksir Hasan Majumder), mmr2512@gmail.com (Md. Mahabur Rahman), anindya@cse.buet.ac.bd (Anindya Iqbal), msrahman@cse.buet.ac.bd (M. Sohel Rahman)

Definition 6. ***False Negative (FN)*** *Number of mistakes in predicted negative values (i.e., the value of actual class is yes, however the value of the predicted class is no.)*

Definition 7. ***Accuracy*** *The ratio of the number of correctly predicted samples and the number of total samples.*

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

Definition 8. ***Precision*** *The ratio of correctly predicted positive samples to the total predicted positive samples*

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Definition 9. ***Recall*** *The ratio of correctly predicted positive samples to the all samples in actual class.*

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Definition 10. ***F1-Score*** *F1-Score is the weighted average of Precision and Recall.*

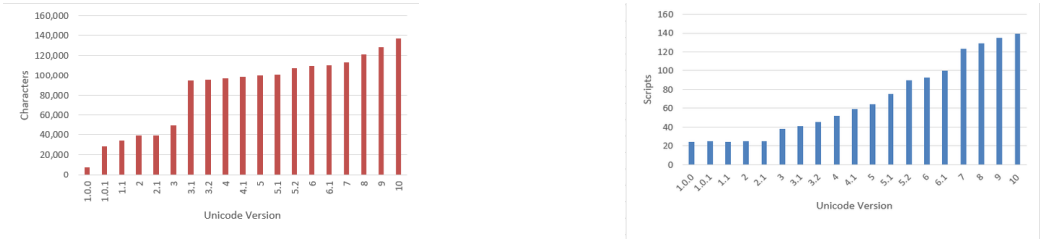
$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

Definition 11. ***Matthews Correlation Coefficient(MCC)*** *The MCC is in a correlation coefficient between the actual and predicted binary classifications. It returns a value between -1 and +1. A coefficient of +1 represents a perfect prediction. A coefficient of 0 means random prediction and -1 means total dissimilarity between prediction and actual labels.*

$$MCC = \frac{TP * TN - TN * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \quad (5)$$

2. Historical Background on Exploiting Homoglyphs

With the globalization of the Internet, utilizing Unicode characters has become a common matter. Unicode consists of a huge set of letters, punctuation marks, mathematical and technical symbols, arrows, emojis, hieroglyphics from a great number of written languages. Although it contains around 136K characters [1], Unicode potentially has the capacity to contain around 1M letters.



a Trend of increase in number of Unicode characters

b Trend of increase in number of Unicode scripts

Figure 1: Status of Unicode progression with each new version release

Now, detecting homoglyphs becomes easier if a list is present with such confusing and visually similar looking characters. However, manually building such a list using only eye-test will be really cumbersome since the number of potential character pair combination is huge. In fact, after the release of Unicode standard version 10.0 in June, 2017, there exists $136,755$ characters [2, 1]. Therefore, a manual inspection may require 10^{10} individual inspections, which is a daunting task. Since the release of Unicode version 1.0

more and more characters with unique identifying names are being mapped to fixed code points in various categories. There are still many scripts that are yet to be included in Unicode. For example, Mayan and RongoRongo scripts are still awaiting approval. Tengwar, being a modern invented script is also awaiting its inclusion [2]. Clearly, the possibility of exploitation of visually similar characters will increase with the expansion of Unicode character set. As a result, with such expansions, probabilities of attacks will rise too. Finding homoglyphs can be seen as the first step to the key defence mechanism against such spoofing attacks. Fig. 1 demonstrates how with each new version, the number of characters as well as the scripts that Unicode contains increases.

Due to Internationalized Domain Names (IDN) and Internationalized Resource Identifiers (IRI), it is now possible to use characters from many languages. A person not only can use an uncommon language in the domain space but also can use letters outside ASCII in web pages, email addresses, and contents. Such exposure to IDNs opened up the path to IDN homograph attacks [3] (which are also done by exploiting homoglyphs). In an IDN homograph attack, a user is deceived by being presented with a URL link, which contains one or more replaced characters, while looking similar [4]. A casual eye can not distinguish the fake address, and thus falls into the trap. For example, a regular user may click a link, `https://apple.com`, where all the characters in the link come from Cyrillic script instead of Latin. Also, another example is replacing the Latin 'a' (U+0061) with the Cyrillic 'а' (U+0430) in "`PAYPAL.com`" and thereby redirecting the user to a fake PAYPAL website for stealing confidential credentials once she clicks the link.

A real-world phishing incident was reported in Austria located at `www.ba-ca.com` by registering the fake domain `ba-cq.com`, where the target was to exploit the visual similarity between the two characters 'a' and 'q'. The target was the Bank Austria Web site. Since, it is a major financial institution, phishing such a site can easily allow a malicious user to steal millions of dollars from naive Internet users. Clearly, a homoglyph based attack could be even more harmful.

3. *Transfer Learning Architectures:*

Concepts of several of the architectures that are used in pre-trained models (in the context of transfer learning) are presented in the following subsections:

3.1. *VGG-16 and VGG-19 Network Architecture*

VGG-16 [5] network consists of 16 convolutional layers in a flat uniform style. The layers use 3×3 filters only, with 2×2 size maxpooling layers in between them. Fully connected layers with softmax activation at the end complete the network. First few layers learn the low-level features (such as edges, shapes, colors), while the remaining layers gradually learn the high-level ones. Figure 2 shows the layer-stacking property of the architecture of the VGG-16 network. VGG-19 network architecture follows the same rule, with the difference being in the number of layers.

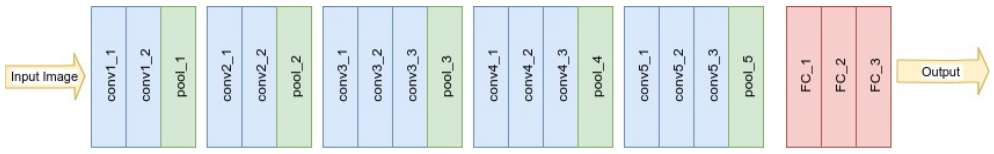
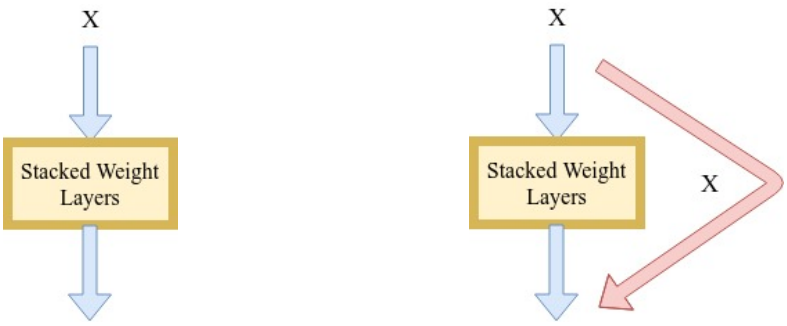


Figure 2: VGG-16 Network Architecture

3.2. ***ResNet Network Architecture***

Deeper networks do not always relate to better performance. In fact, with the increase in network depth, accuracy eventually starts to saturate and degrades quickly [6]. The authors of [6] proposed a hypothesis that direct mapping is hard to approximate. They proposed that instead of training a network to learn a direct mapping from x to $H(x)$, it is better to make it learn the residual between the two, which is $H(x) - x$. Figure 3 demonstrates the intuition behind a residual block in ResNet. Such an architecture allows CNN models to be deeper than usual by stacking lots of layers with residual blocks.



a $F(x) = y$ direct mapping is hard, therefore identity mapping, $y = x$ becomes hard to achieve.

b $F(x) = 0$ mapping is easier compared to $F(x) = y$ direct mapping, therefore identity mapping, $y = x$ becomes easier.

Figure 3: Identity mapping in a Residual Block of ResNet

3.3. ***Inception Architecture***

Inception architecture [7] also relies on stacking layers. While such stacking leads to more depth, this architecture is also wider. Each module recognizes features at multiple scale length, i.e., filters of different sizes in the stacked modules. In order to tackle high computational cost, 1×1 size filters are usually used for dimensionality reduction. As a result, training such

architectures is even less expensive than the VGG-16 architecture. Figure 4 shows how an inception module may look like.

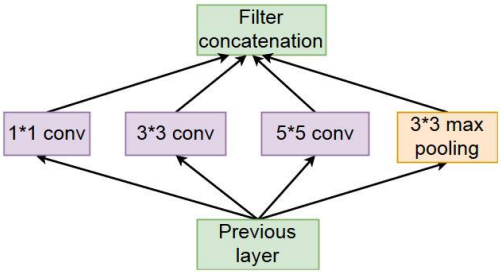


Figure 4: Inception Architecture

3.4. ***Additional Detail about the Dataset***

According to our problem formulation, the model requires to take two different images as input. We generated the character images in such a way that each image has a dimension of 75×75 pixels with three channels. Most of the images had the characters in the center, however several of them were slightly dislocated from the center. The input to our neural networks is fed as a concatenation of a pair of images. Therefore, the input to the neural networks can be thought of as a single 75×75 pixels image. Our models do not impose any restrictions with respect to the number of channels in the input image pair, i.e., given a dataset with colour patches the networks could be trained to further increase performance. We chose to use gray-scale images. In all cases, the characters have been drawn in *black* color over a *white* background.

Our dataset was split into training and validation datasets in a careful manner. However, the independent test dataset was chosen randomly to avoid any specific bias. If a particular character appears in both training

and validation dataset as part of some confusable character pairs, but never appears as part of some non-confusable character pairs therein, it is possible that the neural network will learn its features to predict a confusable character pair whenever that character belongs to the character pair to be predicted. For example, suppose in the confusable character pair set, the letter *'k'* appears more frequently than other characters (that is, there are many characters that are visually similar to *'k'*). What may happen is that a neural network with great capacity may learn a feature that, the presence of *'k'* in any of a character pair means that the characters in the pair are visually similar, which is absolutely not acceptable. As a result, we carefully built the dataset in a way so that such cases do not make the model include features in its prediction task which eventually will be useless in detecting *homoglyphs*. We think that it is very important to build a labeled dataset in a cautious manner which assists the model for generalization purposes (i.e., predictive tasks). In our case, we had the freedom to design the dataset of visually dissimilar character pairs.

4. Various Other Attempts

Below we describe some of our earlier experiments that were not upto expectations along with some qualitative discussions. Notably, the early experiments actually provided on the rationale behind finding our first three model architectures.

4.1. ***Selection of optimizers***

We choose Stochastic Gradient Descent (SGD) as the optimizer based on its performance on a series of experiments. We fix all the hyperparameters

except optimizers and we only vary the optimizers. We train the models for a number of times for each of the optimizers and measure the average accuracy for all of them. Figure 5 indicates that although they are almost identical in performance, SGD is slightly better compared to others.

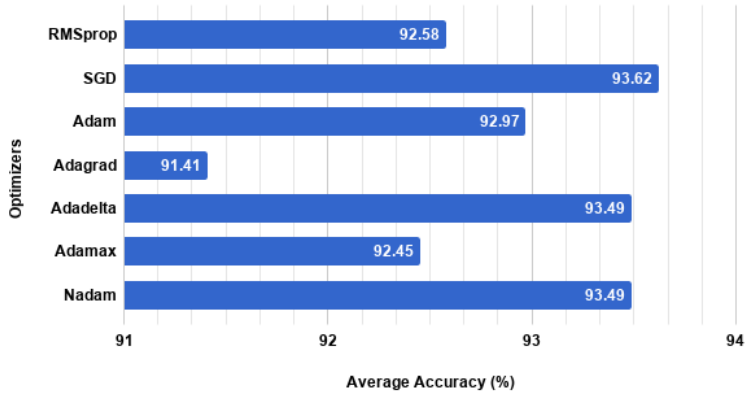
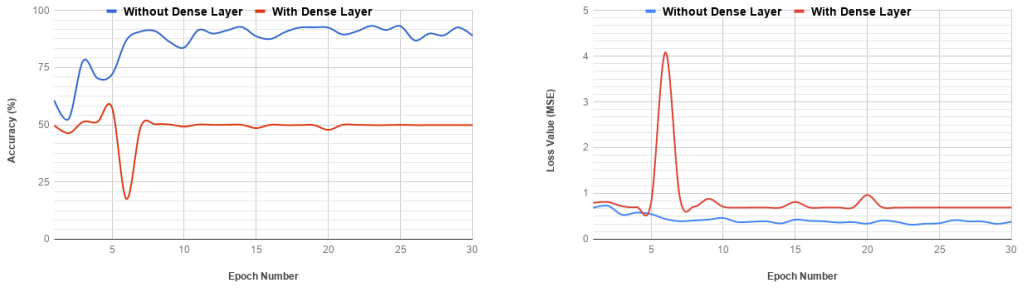


Figure 5: All optimizers perform well, with SGD having a slight edge.

4.2. ***CNN utilizing higher filter size***

When we use filters with 5×5, 10×10 size or higher instead of 3×3 size, performance becomes poor.

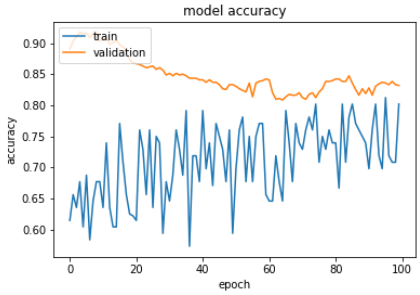


a b

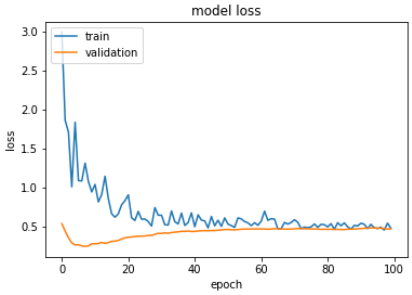
Figure 6: Greater filter size somewhat reduces accuracy. The inclusion of a dense layer reduces it even further, to around 50%, which indicates that the network is learning almost nothing.

4.2.1. ***Transfer Learning with VGG-19 Network***

The VGG-19 architecture is similar to the VGG-16 architecture, however, it has more layers than the former and thus represents a deeper network. The VGG-19 network has an almost identical performance to that of the VGG-16 network. It has 91.80% validation accuracy, which is 4.20% less than the VGG-16 network. Figure 7a and 7b shows the training of the model.



a Optimum validation accuracy value is reached very quickly.



b Loss value of validation starts to diverge early in the training.

Figure 7: Validation accuracy from Transfer Learning from VGG-19 Network reaches upto 91.80%

4.2.2. ***Other pre-trained Networks***

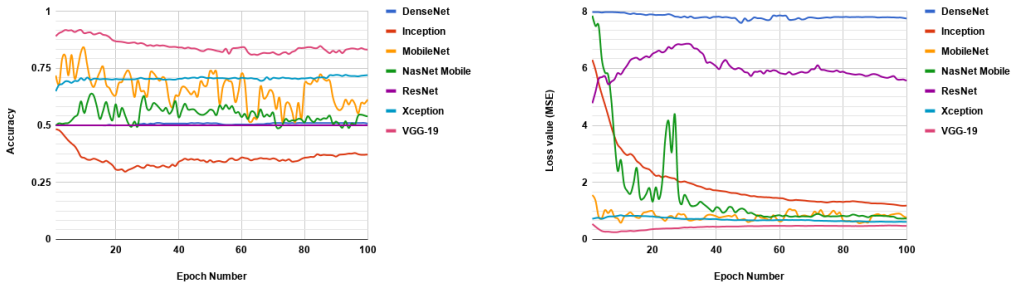


Figure 8: All the architectures display very poor performance compared to that of the VGG-19 network.

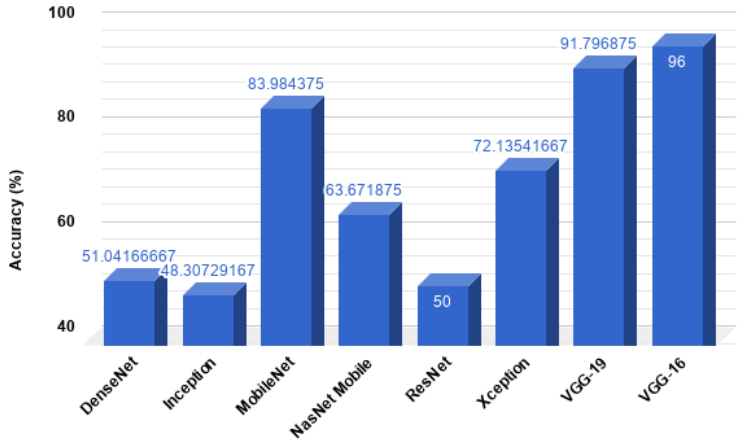


Figure 9: VGG-16 performs best accuracy-wise, followed by VGG-19. All other models fail to perform upto expectation.

All other architectures used in our preliminary experiments have better top-5 (i.e, whether the actual class belongs in the top 5 of the predicted classes) accuracy in the ImageNet dataset than VGG-16 and VGG-19 architecture. Thus, it may appear that feature extraction from these architectures should also result in similar, if not higher, performance in accuracy. However, our experiments demonstrate that they are not suitable for the purpose of classifying homoglyphs and non-homoglyphs. In fact, only mobilenet manages to cross the 80% threshold in our experiments. Xception remains consistent at around 72.14% accuracy. ResNet, NasNet Mobile, Inception and DenseNet- all four of them have only achieve an accuracy closer to 50%. Since this is a binary classification with equal number of training and validation samples, 50% can be least possible accuracy and hence is the baseline for our experiments. Figure 8 reports the training of the models.

Our experiments require networks to learn all types of edges and shapes

as well partial symmetry. These architectures are clearly unable to learn such features, whereas VGG-16 and VGG-19 shine despite their simplistic architecture compared to the rest. Figure 9 presents the comparison of the accuracy metric of the all the models that we leveraged for transfer learning.

5. Brief Discussion

The purpose of transfer learning is knowledge transfer across different domains or tasks [12]. Heterogeneous transfer learning, as it is known, is characterized by the source and target domains having different feature spaces, label spaces, and even dissimilar probabilistic distribution of data. Here, our source domain is ImageNet dataset, which helped in lots of computer vision problems [13, 14, 15, 16]. The target domain is the classification of Unicode character image pairs on the basis of their similarities. We discover in our experiments (discussed in previous sections) that despite the apparent heterogeneity transfer learning can still provide enough pre-trained features for our target domain from the source domain. The experimental results show that transfer learning adoption can achieve strong homoglyph detection capability.

The problem of finding relevant fonts that can render many of the scripts of the Unicode refrained us from producing image data for many of the rarely used Unicode characters. Moreover, we decided to leave out the *private use area* [17] code characters since they are third party dependent and are not under any obligation by the Unicode consortium for quality assurance. Deep learning, in general, requires a lot of data samples in case of high dimensional or complex data form. Obtaining good accuracy in classification problems are

in general difficult when a limited number of labelled training examples are provided [18]. Both $CNN|^{1}_{Scratch}$ and $CNN|^{2}_{Scratch}$ had very high accuracy. It is expected that with a larger dataset the model performance will be even better.

Normalization techniques such as batch-normalization and dropout were used to prevent overfitting. We do not want our models to associate too much with the features of the characters (that appear more frequently in training examples) as high-level features and look for them during the classification task. To overcome overfitting (that is, having a higher training accuracy than validation accuracy), we adopted aggressive regularization measures so that it becomes hard for the network to memorize more frequent characters, and to learn and use their features for predictive tasks.

6. Future Research

Google has a project that aims to unify all Unicode characters under a single font family, known as Noto sans family [19]. The project is still halfway but is progressing at a good pace. We aim to use the full Noto sans font family to negate the effect of lack of fonts for many scripts and extend our current work. It will allow us to increase our labeled dataset where every image rendered by fonts will have a unified theme. After we obtain the full Unicode character dataset, we expect to build a better model. The model along with its weights will be released for public use.

Another interesting avenue would be to use generative adversarial networks (GAN) [20] to artificially create many more characters and augment the available dataset size for the homoglyph dataset.

Siamese Network is another interesting neural network architecture for similarity metric calculation, which has been applied to many interesting problems [21]. We plan to extend our research with Siamese network-based models.

7. False Positives by $CNN|_{Xfer}$

Table 1: These are the scripts that the false positive prediction outcomes of $CNN|_{Xfer}$

Script of the first glyph	Script of the second glyph	No. of false positives
CJK Unified Ide.	CJK Compatibility Ide.	2
CJK Unified Ide.	CJK Radical Supplement	5
CJK Compatibility Ide.s	CJK Compatibility Ide.	2
CJK Compatibility Ide.	Kangxi Radicals	1
CJK Compatibility Ide.	CJK Compatibility Ide.	2
CJK Compatibility Ide.	CJK Radical Supplement	1
CJK Radical Supplement	CJK Radical Supplement	1
CJK Radical Supplement	Kangxi Radicals	1
CJK Com. Ide. Sup.	CJK Radical Supplement	2
CJK Radical Supplement	Khmer	1
Khmer	Kangxi Radicals	1
Arrows	Miscellaneous Technical	1
Hangul Compatibility	Jamo Katakana	1
Khmer	Spacing Modifier Letters	1
Tamil	Katakana	1
Vertical Forms	Hangul Compatibility Jamo	1
Vertical Forms	U.C.A.S.	1
Vertical Forms	Khmer	1

[1] Unicode consortium, version 10.00, http://www.unicode.org/versions/Unicode10.0.0/, online; accessed 16 September 2018 (2018).

[2] Unicode article in wikipedia, https://en.wikipedia.org/wiki/Unicode/, online; accessed 16 September 2018 (2018).

[3] E. Gabrilovich, A. Gontmakher, The homograph attack, Communications of the ACM 45 (2) (2002) 128.

[4] O. J. Hunt, I. Krstic, Preventing url confusion attacks, uS Patent 9,602,520 (Mar. 21 2017).

[5] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Computer Vision and Pattern Recognition (CVPR), 2015.
URL `http://arxiv.org/abs/1409.4842`

[8] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic, Deep learning applications and challenges in big data analytics, Journal of Big Data 2 (1) (2015) 1.

[9] C. C. Charalambous, A. A. Bharath, A data augmentation methodology for training machine/deep learning gait recognition algorithms, arXiv preprint arXiv:1610.07570.

[10] F. Chollet, et al., Keras, `https://github.com/fchollet/keras` (2015).

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467.

[12] K. Weiss, T. M. Khoshgoftaar, D. Wang, A survey of transfer learning, Journal of Big Data 3 (1) (2016) 9.

[13] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, R. M. Summers, Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning, IEEE transactions on medical imaging 35 (5) (2016) 1285–1298.

[14] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient transfer learning, arXiv preprint arXiv:1611.06440.

[15] D. Mehta, H. Rhodin, D. Casas, O. Sotnychenko, W. Xu, C. Theobalt, Monocular 3d human pose estimation using transfer learning and improved cnn supervision, arXiv preprint arXiv:1611.09813.

[16] W. Zhi, H. W. F. Yueng, Z. Chen, S. M. Zandavi, Z. Lu, Y. Y. Chung, Using transfer learning with convolutional neural networks to diagnose breast cancer from histopathological images, in: International Conference on Neural Information Processing, Springer, 2017, pp. 669–676.

[17] Private use area (pua), online; accessed 16 September 2018 (2018). URL `https://en.wikipedia.org/wiki/Private_Use_Area/`

[18] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[19] Google noto sans font family), online; accessed 16 September 2018 (2018).
URL https://www.google.com/get/noto/

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[21] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. Torr, Fully-convolutional siamese networks for object tracking, in: European Conference on Computer Vision, Springer, 2016, pp. 850–865.