Enhancing image classification with transfer learning: An evaluation of overfitting reduction techniques on CINIC-10 dataset

Aditya D Pandey
The University of Adelaide
Adelaide, South Australia, Australia
aditya.pandey@student.adelaide.edu.au

Abstract

Image classification is a fundamental task in computer vision, and it is an essential task in the field of computer vision. In this paper, we evaluate overfitting reduction techniques along with transfer learning with traditional machine learning(ML) approach, and we use CINIC-10 dataset for our task. We use transfer learning to re-use a training model from a task to improve performance. Pre-trained Visual Geometry Group 16 (VGG) model which has been trained on ImageNet dataset. This model achieves the best validation accuracy of 74.41 % using traditional ML approach, while the transfer learning approach offers best accuracy of 53.09%.

1. Introduction

Image classification, a fundamental task in computer vision, involves assigning predefined categories to images. The advent of deep Convolutional neural networks (CNNs) have significantly improved the performance of image classification tasks. CNNs were inspired by biological processes described by Hubel and Wiesel [1], early work by Fukushima on self-organizing neural network for pattern recognition is regarded as the precursor of CNNs [2]. Concepts such as shift invariance and backpropogation [3], max pooling [4] and gradient descent [5] were essential to success of CNN as a whole. Further development may be attributed to Large Scale Visual Recognition challenges such as ImageNet [6], which motivated towards development of several CNN architectures such as AlexNet [7], ZFNet [8], VGGNet [9], GoogleNet [10], Resnet [11], MobileNet [12] just to mention some. The dataset complexity and the task at hand often dictates the ease of prediction, and the dataset used in this paper is CINIC-10, previous studies to perform image classification on the dataset using simple CNN with 4 Convolutional layer achieved a classification accuracy of 92.21 % [9], Simonyan and Zisserman also used ResNet20 V1 and ResNet29 V2 to have 74.91 and 78.28 % accuracy respectively. The Xception model achieved an accurarcy of 85.87% [14]. The SOTA accuracy of 95.064 attained by EfficientNet[15]. We use them as reference for our comparative analysis.

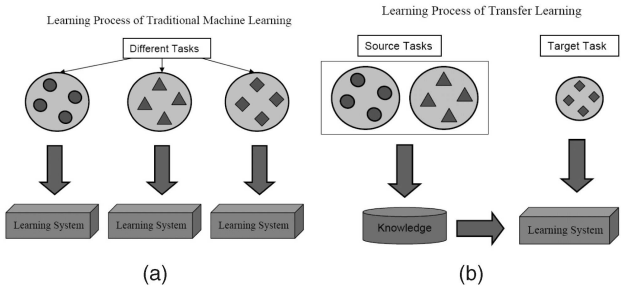

Figure 1. Different learning processes between (a) traditional machine learning and (b) transfer learning. [13]

In this paper we utilize transfer learning, which can be best described as a machine learning model technique where a learning model from a task is re-used in order to boost performance on a related task[16]. We use a network-based deep transfer learning method; it reuses the partial pre-trained network in the source domain, and transfers it to be a part of neural network in target domain [17].

2. Method

In this section we will delve into the dataset and methods utilized in the research study. Subsequently, we will provide a breakdown of the dataset, methods and approach, in subsequent subsections.

2.1. Dataset

We use the CINIC-10 dataset for our classification task, which is an augmented extension of CIFAR-10 by combining with images downsampled from ImageNet dataset [18]. CIFAR 10 is considered to be the goto dataset for understanding and starting with Image Classification [19], but it

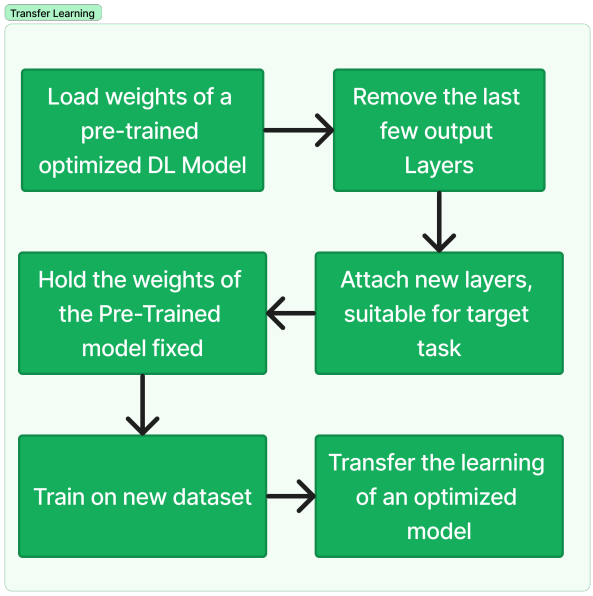

Figure 2. Flowchart for Transfer Learning

is often considered small and straightforward, on the other hand bigger datasets such as like ImageNet32 [20] and Imagenet64 [21] offer challenge in terms of scale and complexity of 100 classes [6]. In this context CINIC-10 ('CINIC 10 Is Not ImageNet or CIFAR 10') [18] emerges as a solution. It expands the CIFAR 10 dataset by incorporating sampled images from ImageNet. CINIC 10 offers features, a large collection of 270,000 images, which is approximately 4.5 times the size of CIFAR 10. Image dimensions standardized to match those of CIFAR 10 for easy integration and an equal distribution of images into training, validation and test sets. Each subset consists of 90,000 images, distributed across ten classes, with 9,000 images allocated to each class in every subset. It allows for potential combinations of training and validation subsets to create training datasets. Furthermore, CINIC-10 provides a unique challenge by featuring images from both CIFAR and ImageNet.

2.2. Transfer Learning

In this work, we utilize the already trained Model along with the weights. Contemporary CNNs, trained on extensive visual datasets have been observed to develop comparable hierarchies of features [22]. Transfer Learning offers the opportunity to utilized the pre -trained features and is considered to be helpful for training even on a target domain which might be visually dissimilar to the source domain [23]. To analyze this ability of feature reuse, we use a source domain comprising of natural images, Imagenet [6], and a the target domain of CINIC-10 [18].

The definition of 'domain' and 'task' are taken from works of Pan and Yang. A domain, denoted as D_T , encompasses a feature space X and a marginal probability distribution $P(X)P(X)$, with X being a set of features $(X = x_1, ..., x_i)$. To illustrate, think about the task of document classification where each term serves as a binary feature. In this context, X represents the vector space for all terms, and x_i stands for the term vector associated with specific documents. It is worth noting that if two domains are distinct, they tend to exhibit differences in their feature spaces and marginal probability distributions. Domain, $D = X, P(X)$ (with features X and marginal probability distribution $P(X)$), a task comprises two key elements: a label space y and an objective predictive function $f : X \rightarrow Y$. This task is represented as $T = Y, f()$. The predictive function is acquired through learning from training data, which consists of pairs x_i, y_i where x_i belongs to X and y_i belongs to y, enabling the prediction of new labels $f(x)$ for a given instance x. By definition transfer learning is 'Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S , where $D_S \neq D_T$, or $T_S \neq T_T$ '[17] [13] [24]. The main issues to consider for Transfer Learning:

- What to transfer
- How to transfer
- When to transfer

We address 'What to transfer' by identifying which source knowledge is applicable and beneficial for enhancing target domain or task performance. 'How to transfer' involves selecting a suitable learning algorithm for knowledge transfer, while 'when to transfer' deals with the timing of knowledge transfer. It is important to note that discerning 'when not to transfer' is as crucial as knowing 'when to transfer' [25].

2.3. Model Pipeline

We use the transfer learning techniques as depicted above in Figure 2 using the VGG16 model [9], also called as the OxfordNet named after the Visual Geometry Group from Oxford as our base model. Utilizing it, we define 4 approaches to train them on the dataset. The pre-trained weights of the VGG16 model are used to perform feature extraction on CINIC-10 dataset. In model 1, the classification layer of VGG model is replaced with two dense layers which have Relu activation, and the output layer with softmax activation function. In model 2, we introduce data augmentation to train the model, with the same architecture as model 1. We perform data augmentation process on the training images using ImageDataGenerator from Keras library with augmentations such as Random rotation, random height shift, random shearing transformation, zooming, horizontal flips which is the standard solution against

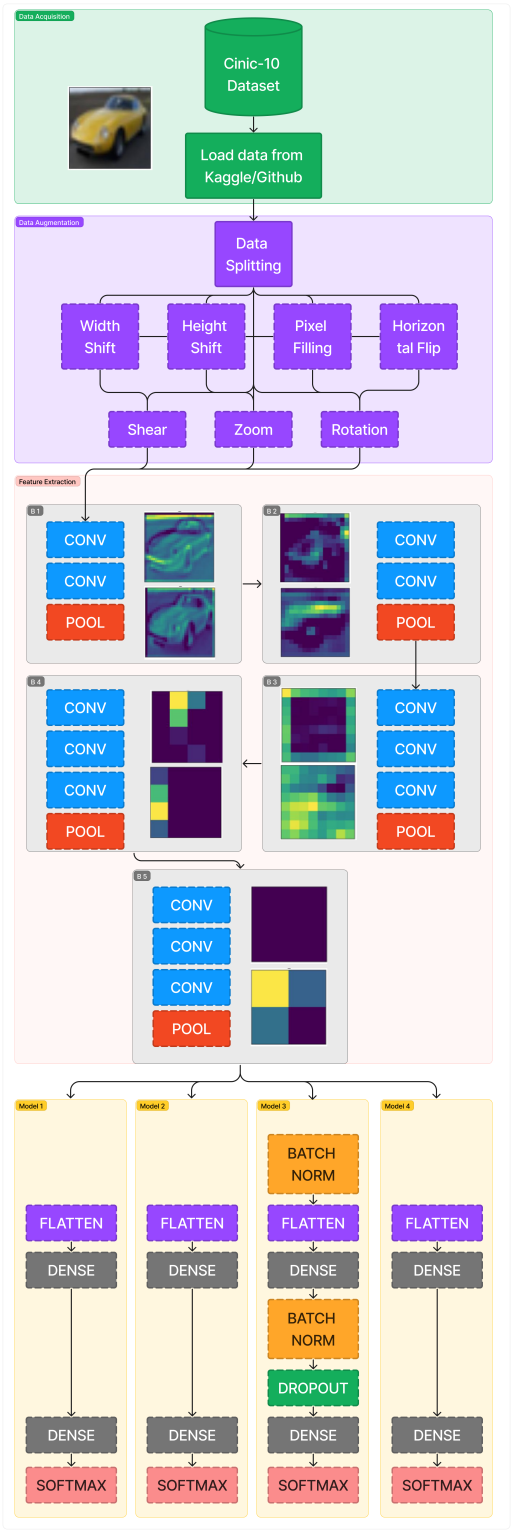

Figure 3. Flowchart for Deep Learning Pipeline

over-fitting [26]. In model 3, we implement regularization utilizing dropout, batch normalization layers to obtain ideal classification performance and to reduce over-fitting [27]. In model 4, we use the same model architecture as model 1 and make the complete model trainable. We make the convolutional layers of model 1, 2 and 3 untrainable in order to retain the pre-trained weights of the VGG16 model. The architecture of the model we use are depicted in Figure 3 and in much more detail in figure 4 and 5. The knowledge from the model is transferred in form of the weights which were generated when it was trained on classification of 1000 on 14 million images. The feature extraction results are then fed to the dense layer blocks. We use Adam optimiser to update the model weights which can be described as a stochastic gradient descent method with adaptive estimator of lower-order moments with an adaptive learning rate[28]. It has been proven to be suitable and computationally efficient for large scale data [28]. The process of training and evaluation is described in Figure 3 Flowchart of the Deep Learning Pipeline. The first step is Data Acquisition, we download the data locally, or for a implementation on platforms such as Google Collab and Kaggle which offer GPU compute for free, we utilized kaggle dataset for the same. The flowchart depicts an image from the CINIC-10 Training Directory of a car here. We then apply the data augmentation as described in the earlier parts; all the augmentations are random in nature with a degree as is specified in the code. We then pass the image through the pre-trained VGG model and there are images that depict the feature extraction from them for there respective blocks. We replace the classifier section of the VGG net as described above, and is depicted in Figure 3 and 4. After training of the model, we save it and test it on the testing set of images and determine the accuracy, precision and recall to ascertain the performance of the model. We use learning rate of 0.001 and loss is based on Categorical Cross Entropy, we set the epoch limit to 55 but along with that we implement an early stopping call-back [29] with patience level of 10 and monitor the validation loss, which would stop the training if the validation loss tends to not decrease for 10 consecutive epochs which prevents the model from over-fitting. We train the model until the early stopping criteria is hit. We use the retraining feature of Keras to save the model, and restart the training from the point were it was stopped. For model 3 and 4 the model were trained uptill epoch 71 and 82 respectively. While models 1 and 2 trigerred the early stop at epoch 14 and 32 itself respectively.

2.4. Experimental Analysis

We evaluate the model on the test set of CINIC-10 with 9000 images for each class of the dataset. We use the trained models on the test set and calculate the Accuracy. We evaluate the training of the models based on Validation Accuracy

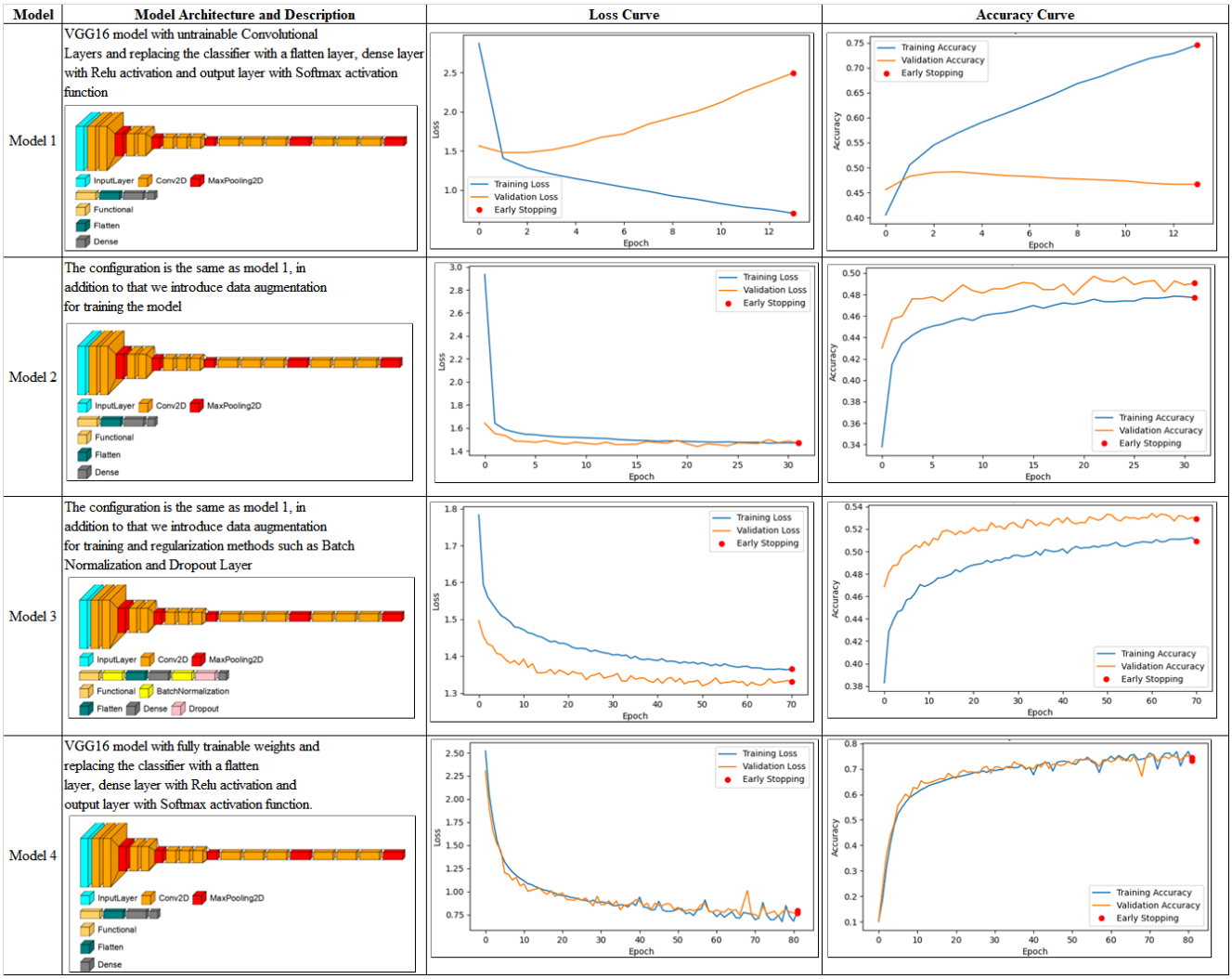

Figure 4. Architecture, description, loss and accuracy curves of each model

and Loss.

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

In the above equation TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative. TP (TN) represents the number of observations in the positive (negative) class that are classified as positive (negative), and FP (FN) represents the number of observations in the negative (positive) class that are classified as positive (negative)[30].

In Figure. 4, the second column defines the model description and architecture, it is separated in two parts in order to specify the VGG16 model and second part with the custom layers which are added to model for this paper. In second and third column we plot the training and validation loss and accuracy curve for each model.

We notice that in model 1 the early stopping is triggered much earlier than other models, implying overfitting; the validation loss was on an upward trend while the training loss seemed to be decreasing and vice versa for the accuracy counterparts. Model 2 introduces image augmentation and has a better training compared to model 1, with both graphs have converging curves for training and validation curves and the model trains for much longer than model 1 too. In model 3, regularization layers are added to the model and it is evident it helps in attaining a comparatively better accuracy, the reduction in validation loss continues till epoch 71 until early stopping is triggered. Model 4 we train all layers of the model with them initialized to the pre-trained weights. This model attains the best validation accuracy of 74.41 by the end of training and trains till epoch 82. We tabulate the classification accuracy and the number of trainable parameters of each model along with comparison among models which achieve far greater validation accu-

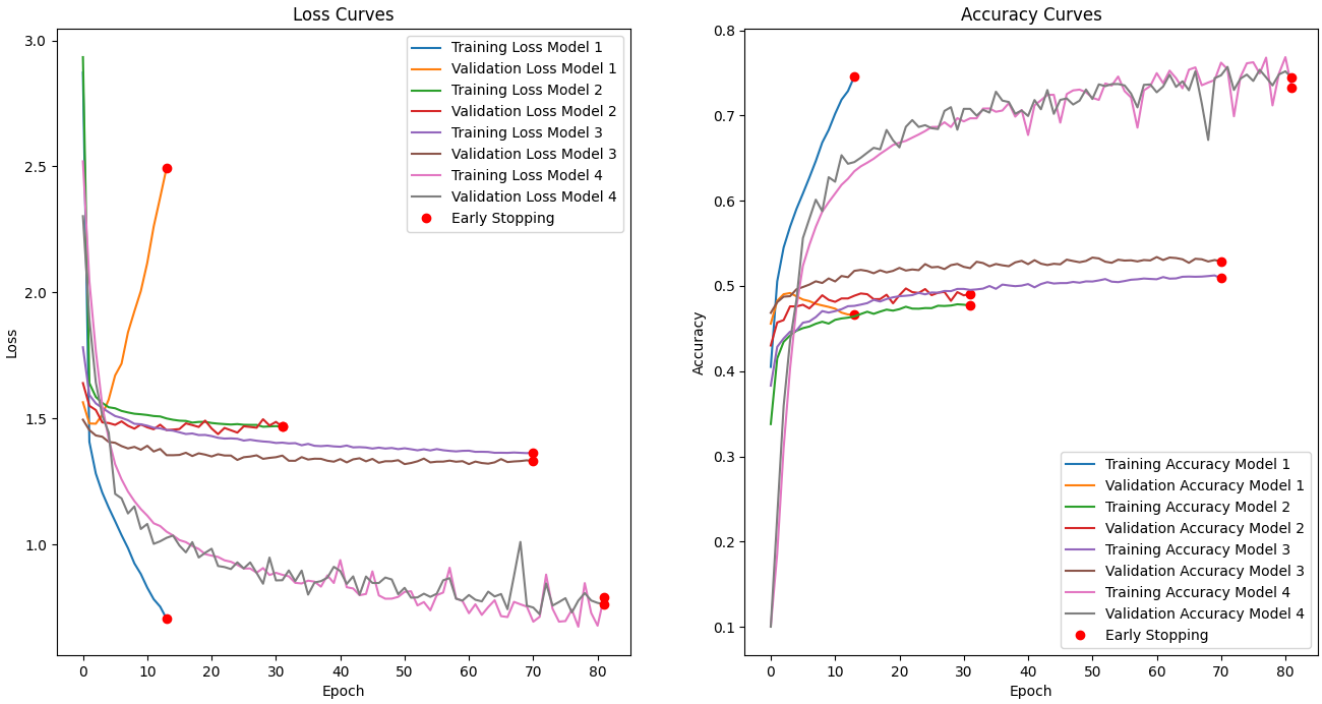

Figure 5. Combined Loss and Accuracy curve of model 1-4

Model	Classification Accuracy (%)	Number of Trainable Parameters
Model 1	48.7	2,67,786
Model 2	49.22	2,67,786
Model 3	53.09	2,69,834
Model 4	75.27	1,49,82,474
CNN with 4 Convolutional Layer[31]	92.21	28,17,042
ResNet20 V1[31]	74.91	2,74,442
ResNet29 V2[31]	78.28	8,49,002
Xception[14]	85.87	2,28,55,952
EfficientNet[15]	95.064	1,10,00,000

Table 1. Model Comparison

racy without transfer learning. Transfer learning approach use far lower number of trainable parameters but still fail to provide comparable classification accuracy.

2.5. Conclusion and Future Work

This paper is intended to evaluate the overfitting reduction techniques and transfer learning, we observe a consistent improval of validation accuracy across each model,as techniques such as Early Stopping, Data Augmentation and Regularization are introduced to the model, it is evident they help in reducing overfitting and help in training the model for more epochs with much better validation accuracy.Transfer learning approach underperforms than traditional machine learning approach for CINIC-10 dataset with the best accuracy of 53.09% while the traditional machine learning approach with no overfitting reduction techniques obtained the accuracy of 75.27 %. Though Transfer learning offers the prospect of reduced computational load and re-utilization of pre-trained models, it does not provide comparable accuracy. There is scope of fine tuning the model based on the dataset in order to achieve good classification accuracy.

2.6. Code

My code and data is available at `https://github.com/aditya-524/DL_Ass2/blob/main/Keras%20Approach/VGG_4_models.ipynb`. The code utilizes the python packages as such matplotlib [32], numpy [33], Tensorflow & Tensorflow-Keras [34], VisualKeras [35]. Official tutorial from Keras Transfer Learning Guide was used as reference [36].

References

[1] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[2] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[3] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[4] Kouichi Yamaguchi, Kenji Sakamoto, Toshio Akabane, and Yoshiji Fujimoto. A neural network for speaker-independent isolated word recognition. In *ICSLP*, 1990.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks, 2017.

[8] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520. IEEE, 2018.

[13] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[14] Haiying Yuan, Junpeng Cheng, Yanrui Wu, and Zhiyong Zeng. Low-res mobilenet: An efficient lightweight network for low-resolution image classification in resource-constrained scenarios. *Multimedia Tools and Applications*, 81(27):38513–38530, 2022.

[15] Bruno Antonio, Davide Moroni, and Massimo Martinelli. Efficient adaptive ensembling for image classification. *Expert Systems*, 2022.

[16] Jeremy West, Dan Ventura, and Sean Warnick. Spring research presentation: A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, 1(08), 2007.

[17] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, volume 11141 of *Lecture Notes in Computer Science*, pages 270–279. Springer International Publishing, Cham, 2018.

[18] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[21] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[22] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[23] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

[24] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of big data*, 3(1), 2016.

[25] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[26] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 558–567, 2019.

[27] Haibing Wu and Xiaodong Gu. Max-pooling dropout for regularization of convolutional neural networks. In *Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part I 22*, pages 46–54. Springer, 2015.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural networks*, 11(4):761–767, 1998.

[30] Leila Ismail, Huned Materwala, Maryam Tayefi, Phuong Ngo, and Achim P Karduck. Type 2 diabetes with artificial intelligence machine learning: methods and evaluation. *Archives of Computational Methods in Engineering*, pages 1–21, 2022.

[31] Mohsin Sharif, Asia Kausar, JinHyuck Park, and Dong Ryeol Shin. Tiny image classification using four-block convolutional neural network. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1–6. IEEE, 2019.

[32] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science engineering*, 9(3):90–95, 2007.

[33] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature (London)*, 585(7825):357–362, 2020.

[34] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[35] Paul Gavrikov. visualkeras. `https://github.com/paulgavrikov/visualkeras`, 2020.

[36] François Chollet. Keras documentation: Transfer learning fine-tuning. `https://keras.io/guides/transfer_learning/`, Apr 2020.