A Study on CNN Transfer Learning for Image Classification

Mahbub Hussain, Jordan J. Bird, and Diego R. Faria(✉)

School of Engineering and Applied Science, Aston University, Birmingham B4 7ET, UK
{hussam42,birdjl,d.faria}@aston.ac.uk

Abstract. Many image classification models have been introduced to help tackle the foremost issue of recognition accuracy. Image classification is one of the core problems in Computer Vision field with a large variety of practical applications. Examples include: object recognition for robotic manipulation, pedestrian or obstacle detection for autonomous vehicles, among others. A lot of attention has been associated with Machine Learning, specifically neural networks such as the Convolutional Neural Network (CNN) winning image classification competitions. This work proposes the study and investigation of such a CNN architecture model (i.e. Inception-v3) to establish whether it would work best in terms of accuracy and efficiency with new image datasets via Transfer Learning. The retrained model is evaluated, and the results are compared to some state-of-the-art approaches.

1 Introduction

In recent years, the field of Machine Learning has made tremendous progress in different domains where autonomous systems are needed. Thus, allowing to advance models such as a Deep Convolutional Neural Networks to achieve impressive performance on hard visual recognition tasks, matching or exceeding human performance in some domains [1]. The work, "Going Deeper with Convolutions" [2] introduces the Inception-v1 architecture, which was well succeeded in the ILSVRC 2014 GoogleNet challenge. The main contribution presented by the authors is the application to the deeper nets required for image classification. The authors observed that some sparsity would be beneficial to the network's performance, and thus it was applied using today's computing techniques. The authors also introduced additional losses to help improve convergence on the relatively deep network. The limitations noted was a training trick, which also resulted in the output of the layers being discarded during inference. The authors in [3] propose a novel deep network structure to help the enhancement of a model that distinguishes between patches in the receptive field of a convolutional layer within a CNN by instantiating a micro network using multilayer perceptron's instead of linear filters and non-linear activation functions to abstract the data. Similarly to a common CNN, micro networks stride over input images to produce a feature map, these types of layers can then be stacked resulting in a deep "network in network". The results presented within that paper found that the proposed network was less prone to overfitting than traditional fully connected layers due to a global average pooling over

A. Lotfi et al. (Eds.): UKCI 2018, AISC 840, pp. 191–202, 2019.
https://doi.org/10.1007/978-3-319-97982-3_16

the feature maps in the classification layer. Tests were conducted on numerous datasets, one of which was CIFAR-10 [10]. Results presented focused on test error rates regarding network in network with combination of a drop-out and data augmentation, which achieved the best scores. Their limitation include the usage of multiple layers and combinations, and this dissents the availability of being able to identify and compare singular layers and their effect on performance. The book "Computational collective intelligence" [4], covers various aspects of Machine Learning. The author tests 3 different architectures: a simple network trained on the CIFAR-10 dataset, a CNN trained on the MNIST dataset as well as a CNN trained on the CIFAR-10 dataset. The authors explain the testing procedures undertaken in detail, in terms of number of convolutional layers, max pooling layers and ReLU layers used. The author also details tests evaluating the performance of each architecture in combination with local binary patterns (LBP). The simple network achieved a test accuracy score of 31.54%, with the CNN trained on the CIFAR-10 dataset managing to achieve a higher score of 38.8% after 2805 s of training. Most of the aforementioned papers identified limitations whether it be cost, insufficient requirements or problems with the processing of complex datasets, or quality of images. Rather than replicating these studies, the aim of this work is to progress on previous related works and solve the addressed limitations. The results obtained by the authors in [4], will be used as a direct comparison. We will use a similar architecture and dataset compared to [4], however combining Transfer Learning on top of a pre-trained model on other datasets (domains). The aim is to establish whether accuracy results could be improved with Transfer Learning given minimal time and computational resources.

Therefore, the motivation of this work encompasses finding a suitable model in which facilitates Transfer Learning, allowing classification of new datasets with respectable accuracy. The main contributions of this work are as follows: a study on the core principals behind CNNs related to a series of tests to determine the usability of such as technique (i.e. Transfer Learning) and whether it could be applied to multiple datasets with different images category. Also, acknowledging the measures taken to adapt a network to advance its integration within diverse domains. Thus, we test a CNN architecture (i.e. Inception-v3) on both the Caltech [11] Face dataset and the CIFAR-10 dataset [10] whilst changing certain parameters to evaluate their significance with regards to the classification accuracy results.

The remaining structure of this paper is as follows. The approach adopted in this work is explained in Sect. 2, with the experimental setup and datasets described in Sect. 3, followed by a preliminary set of results explained in Sect. 4. This paper is concluded and then future works based on the findings is proposed in Sect. 5.

2 CNN Transfer Learning Development

2.1 CNN

Convolutional Neural Networks (CNN) have completely dominated the machine vision space in recent years. A CNN consists of an input layer, output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional

layers, pooling layers, fully connected layers and normalisation layers (ReLU). Additional layers can be used for more complex models. Examples of a typical CNN can be seen in [5] and it is depicted in Fig. 1.

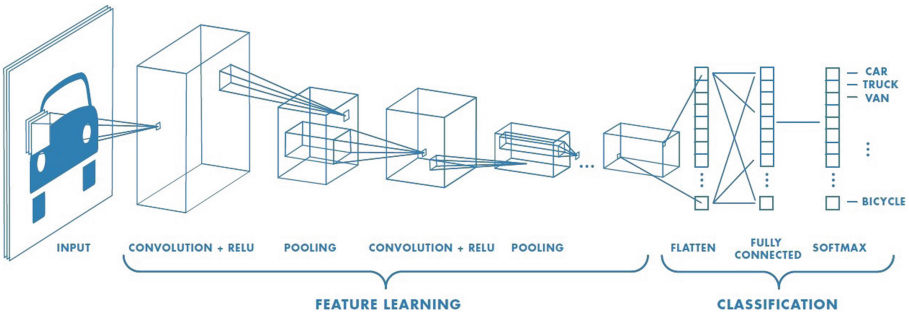

Fig. 1. Typical CNN architecture [14].

The CNN architecture has shown excellent performance in many Computer Vision and Machine Learning problems. CNN trains and predicts in an abstract level, with the details left out for later sections. This CNN model is used extensively in modern Machine Learning applications due to its ongoing record breaking effectiveness. Linear algebra is the basis for how these CNNs work. Matrix vector multiplication is at the heart of how data and weights are represented [12]. Each of the layers contains a different set of characteristics for an image set. For instance, if a face image is the input into a CNN, the network will learn some basic characteristics such as edges, bright spots, dark spots, shapes etc., in its initial layers. The next set of layers will consist of shapes and objects relating to the image which are recognisable such as: eyes, nose and mouth. The subsequent layer consists of aspects that look like actual faces, in other words, shapes and objects which the network can use to define a human face. CNN matches parts rather than the whole image, therefore breaking the image classification process down into smaller parts (features). A 3×3 grid is defined to represent the features extraction by the CNN for evaluation. The following process, known as filtering, involves lining the feature with the image patch. One-by-one, each pixel is multiplied by the corresponding feature pixel, and once completed, all the values are summed and divided by the total number of pixels in the feature space. The final value for the feature is then placed into the feature patch. This process is repeated for the remaining feature patches followed by trying every possible match- repeated application of this filter, which is known as a convolution.

The next layer of a CNN is referred to as "max pooling", which involves shrinking the image stack. In order to pool an image, the window size must be defined (e.g. usually $2\times2/3\times3$ pixels), the stride must also be defined (e.g. usually 2 pixels). The window is then filtered across the image in strides, with the max value being recorded for each window. Max pooling reduces the dimensionality of each feature map whilst retaining the most important information. The normalisation layer of a CNN, also referred to as the process of Rectified Linear Unit (ReLU), involves changing all

negative values within the filtered image to 0. This step is then repeated on all the filtered images, the ReLU layer increases the non-linear properties of the model. The subsequent step by the CNN is to stack the layers (convolution, pooling, ReLU), so that the output of one layer becomes the input of the next. Layers can be repeated resulting in a "deep stacking". The final layer within the CNN architecture is called the fully connected layer also known as the classifier. Within this layer every value gets a vote on determining the image classification. Fully connected layers are often stacked together, with each intermediate layer voting on phantom "hidden" categories. In effect, each additional layer allows the network to learn even more sophisticated combinations of features towards better decision making [6]. The values used for the convolution layer as well as the weights for the fully connected layers are obtained through backpropagation, which is done by the deep neural network. Backpropagation is whereby the neural network uses the error in the final answer to determine how much the network adjusts and changes.

The Inception-v3 model is an architecture of convolutional networks. It is one of the most accurate models in its field for image classification, achieving 3.46% in terms of "top-5 error rate" having been trained on the ImageNet dataset [7]. Originally created by the Google Brain team, this model has been used for different tasks such as object detection as well as other domains through Transfer Learning.

The CNN learning process can rely on vector calculus and chain rule. Let z be a scalar (i.e., $z \in \mathrm{R}$) and $y \in \mathbf{R}^{\mathrm{H}}$ be a vector. So, if z is a function of y, then the partial derivative of z with respect to y is a vector, defined as:

$$\left(\frac{\partial \mathrm{z}}{\partial \mathrm{y}}\right)_i = \frac{\partial \mathrm{z}}{\partial \mathrm{y}_i}. \tag{1}$$

Specifically, $\left(\frac{\partial \mathrm{z}}{\partial \mathrm{y}}\right)$ is a vector having the same size as y, and its *i-th* element is $\left(\frac{\partial z}{\partial y}\right)_i$. Also note that $\left(\frac{\partial z}{\partial y^T}\right) = \left(\frac{\partial z}{\partial y}\right)^T$. Furthermore, presume $x \in \mathrm{R}^{\mathrm{W}}$ is another vector, and y is a function of x. Then, the partial derivative of y with respect to x is defined as:

$$\left(\frac{\partial \mathrm{y}}{\partial \mathrm{x}^T}\right)_{ij} = \frac{\partial \mathrm{y}_i}{\partial \mathrm{x}_i}. \tag{2}$$

This fractional derivative is a H × W matrix, whose entry at the juncture of the *i-th* row and *j-th* column is $\frac{\partial \mathrm{y}_i}{\partial \mathrm{x}_i}$. It is easy to see that z is a function of x in a chain-like argument: a function maps x to y, and another function maps y to z. A chain rule can be used to compute:

$$\left(\frac{\partial z}{\partial x^T}\right), \text{ as } \left(\frac{\partial z}{\partial x^T}\right) = \left(\frac{\partial z}{\partial y^T}\right)\left(\frac{\partial y}{\partial x^T}\right). \tag{3}$$

One can use a cost or loss function to measure the discrepancy between the prediction of a CNN x^L and the target t, $x^1 \rightarrow w^1, x^2 \rightarrow \ldots, x^L \rightarrow w^L = z$, using a simplistic loss function $z = \|t - x^L\|^2$. However more complex functions are usually employed. A prediction output can be seen as $\text{argmax}_i\, x_i^L$. The convolution procedure can be expressed as:

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=0}^{H} \sum_{j=0}^{W} \sum_{d=0}^{D} f_{i,j,d} \times x^L_{i^{l+1}+,i,j^{l+1}+j,d}. \tag{4}$$

The filter f has size $(H \times W \times D^l)$, thus the convolution will have the spatial size of $(H^l - H + 1) \times \left(W^l - W + 1\right)$ with D slices, which means $y\left(x^{l+1}\right)$ in $\mathrm{R}^{H^{l+1} \times W^{l+1} \times D^{l+1}}$, $H^{l+1} = H^l - H + 1, W^{l+1} = W^l - W + 1, D^{l+1} = D$.

When it comes to Inception V3, the probability of each label $k \in \{1, \ldots K\}$ for each training example is computed by $P(k|x) = \frac{\exp(z_k)}{\sum_i^K \exp(z_i)}$, where z is a non-normalised log probability. Ground truth distribution over labels $q(k \mid x)$ is normalised, so that $\sum_k q(k|x) = 1$. For this model, the loss is given by cross-entropy:

$$\ell = \sum_{k=1}^{K} \log(p(k)) q(k). \tag{5}$$

Cross-entropy loss is differentiable with respect to the logits z_k and thus it can be used for gradient training of deep models, where the gradients has the simple form $\frac{\partial \ell}{\partial z_k} = p(k) - q(k)$, bounded between -1 and 1. Usually, when minimising the cross entropy, it means that log-likelihood of the correct label is maximised. Since it may cause some overfitting problems, Inception V3 considers a distribution over labels independent of training examples $u(k)$ with a smooth parameter ϵ, where for a training example, the label distribution $q(k \mid x) = \delta_{k,y}$ is simply replaced by:

$$q'^{(k|x)} = (1 - \epsilon)\delta_{k,x} + \epsilon u(k), \tag{6}$$

which is a mixture of the original distribution $q(k \mid x)$ with weights $1-\epsilon$ and the fixed distribution $u(k)$ with weights ϵ. A label-smoothing regularisation is applied, with uniform distribution $u(k) = 1/K$, so that it becomes:

$$q'^{(k|x)} = (1 - \epsilon)\delta_{k,x} + \frac{\epsilon}{K}. \tag{7}$$

Alternatively, this can be interpreted as cross-entropy as follows:

$$H(q', p) = -\sum_{k=1}^{K} \log(p(k)) q'^{(k)} = (1 - \epsilon) H(q', p) + \epsilon H(u, p). \tag{8}$$

Therefore, the label-smoothing regularisation is similar to applying a single cross-entropy loss $H(q, p)$ with a pair of losses $H(q, p)$ and $H(u, p)$, with the second loss penalising the deviation of the predicted label distribution p from the prior u with

relative weight $\frac{\epsilon}{(1-\epsilon)}$, which is equivalent to computing the Kullback–Leibler divergence. More details (step-by-step) and the mathematical formulation of CNN and Inception V3 can be found in [12, 13].

In this work we aim to retrain this model on a new dataset and study the results. Choosing not to train the model from scratch as it would be computationally intensive task and depending on the computing setup, may take several days or even weeks. In addition, it would also require multiple GPUs and/or multiple machines. Instead, we will be comparing results obtained from the proposed retrained model to that of related works to prove our hypothesis.

2.2 Transfer Learning

Transfer Learning is a Machine Learning technique whereby a model is trained and developed for one task and is then re-used on a second related task. It refers to the situation whereby what has been learnt in one setting is exploited to improve optimisation in another setting [8]. Transfer Learning is usually applied when there is a new dataset smaller than the original dataset used to train the pre-trained model [9].

This paper proposes a system which uses a model (Inception-v3) in which was first trained on a base dataset (ImageNet), and is now being repurposed to learn features (or transfer them), to be trained on a new dataset (CIFAR-10 and Caltech Faces). With regards to the initial training, Transfer Learning allows us to start with the learned features on the ImageNet dataset and adjust these features and perhaps the structure of the model to suit the new dataset/task instead of starting the learning process on the data from scratch with random weight initialization. TensorFlow is used to facilitate Transfer Learning of the CNN pre-trained model. We study the topology of the CNN architecture to find a suitable model, permitting image classification through Transfer Learning. Whilst testing and changing the network topology (i.e. parameters) as well as dataset characteristic to help determine which variables affect classification accuracy, though with limited computational power and time.

3 Experimental Set-Up and Datasets

For the experiments, two datasets are used: CIFAR-10 [10] and Caltech Faces [11] to retrain a pretrained model on ImageNet dataset. Literature review indicated the CIFAR-10 dataset as popular given many researchers would use such a dataset with it having a large set of images that of which were of low dimensionality. The dataset consists of 60000 (32 $\times$ 32 pixel) colour images in 10 classes. The Caltech Face dataset consisted of 450 (896 $\times$ 592 pixels) face images of 27 people. Both datasets vary in terms of image quantity, quality and type, with the CIFAR-10 dataset consisting of several categories (animals, vehicles and ships) whereas the Caltech Face dataset contains only face images. Consequently, allowing us to study the significance of the aforementioned differences with regards to accuracy performance.

With regards to coding, Python is the preferred language, as it not only contained a huge set of libraries that of which were easily used for Machine Learning (i.e. TensorFlow), but was also very accessible. The pre-trained model chosen for Transfer

Learning is the Inception-v3 model created by Google [1], with regards to image data for retraining we used the datasets: CIFAR-10 and Caltech Faces. Using a pre-existing dataset such as CIFAR-10, allows the comparison of results from previous state-of-the-art studies. For training purposes, we have one model retrained on the CIFAR-10 dataset and another model retrained on the Caltech Face dataset. The CIFAR-10 model, consists of 10 classes containing 10000 images. For the testing stage, we will keep it simple and have 2 test images per category, which is sufficient in obtaining preliminary results to meet our aims. It is important to note when training, the images are categorised into different labels identified by folder titles, the CNN will use the labels to classify the test images. When testing, all the images will be placed into one folder without labels. The second model using the Caltech Faces dataset will follow the same procedures, though each of the training classes will comprise of 18 images.

4 Preliminary Results

This section discusses the procedure in setting up tests to evaluate the system as well as documenting the results obtained. The presented tests should help in answering the following questions: Does Transfer Learning help improve the accuracy of a CNN? Does the number of epochs (training steps) improve accuracy? Does the number of images per class in a dataset influence accuracy? Does the type of image in a dataset effect the accuracy? Some of these questions have been influenced having identified open issues within previous state-of-the-art studies described in Sect. 1.

Test 1: The first test involves retraining three Inception-v3 models (pre-trained on the ImageNet dataset) with the datasets: CIFAR-10 and Caltech Faces. CIFAR-10 test A, involves retraining the model with 10000 images per training class, whilst CIFAR-10 test B involves retraining the model with 1000 images per training class. The purpose is to establish whether the quantity of training images affects classification accuracy as well as obtaining the average accuracy achieved by the Inception-v3 model having been retrained on both datasets stated separately.

Figure 2 shows some of the output images having run the CIFAR-10 model. The results from the first test indicated in Table 1 show that the CIFAR-10 test A model achieved a higher average accuracy of 70.1% over the training set compared to the

Fig. 2. CNN Transfer Learning results. Model trained on CIFAR-10 dataset Test A

Table 1. Classification results for both datasets in terms of overall accuracy.

Dataset	Average accuracy (%)
CIFAR-10 (Test A)	70.1
CIFAR-10 (Test B)	66.1
Caltech Faces	65.7

CIFAR-10 test B model and the model retrained on Caltech Face dataset that of which achieved classification accuracies of 66.1% and 65.7% respectively.

The results attained helped in answering the question: Does the number of images per class in a dataset influence accuracy? Results show that the accuracy scores are higher when more sample images are used for training aids. As the CIFAR-10 test B model used the same dataset though with far less training images and thus achieved a lower accuracy percentage, the model which used the Caltech Face dataset also contained less training images compared to CIFAR-10 test A. Furthermore, in terms of training time, the CIFAR-10 test A model took 3 h compared to 30 min for the Caltech trained model. This indicates the more images (although worse in terms of quality) has a significant effect on the training time as well as computational power required.

The results obtained from test 1 are very important towards the motivation of this work, determining whether Transfer Learning is useful in improving accuracy for image classification. Our results evidenced better accuracy scores compared to that achieved in [4] mentioned in Sect. 1. The model presented within this paper achieved almost twice the classification accuracy having been trained on the same dataset (38% vs 70%), with the only difference being the author in [4] had used a CNN-CIFAR-10 model trained from scratch as opposed to a pre-trained model which was adapted using Transfer Learning from ImageNet dataset to be retrained using the CIFAR-10. This evidently shows the benefits of Transfer Learning. Also considering we only used 500 epochs and a CPU due to time and computing power limitations, we could have easily achieved better accuracy scores given the use of GPUs, as they tend to perform a lot quicker and achieve better accuracy scores when more sample images are used for training.

Test 2: For the second test we have changed the number of epochs (full training cycle on the whole training data) to verify whether it influences the overall accuracy. For all tests the default number of epochs was set to 500, though for this test we have used 4000 epochs to replicate the original training of the Inception-v3 by Google on the ImageNet dataset. The training procedure was done on 2 classes consisting of 18 images (from the Caltech Face dataset).

Figure 3 shows the output having run the second test. The images show the input image (testing image) with a text overlaid to indicate the accuracy percentage. The images used for the second test were taken in a good lighting condition and with good angles. The results from the second test presented in Table 2 show that the increase in number of epochs does in fact help with the classification accuracy. With the model using 4000 epochs as opposed to 500 achieving a higher accuracy percentage, though requiring more time to train.

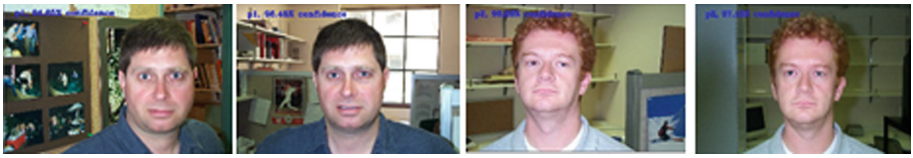

Fig. 3. CNN Transfer Learning results. Model trained on Caltech Faces dataset. Accuracy (confidence) for the images from left to right: 94.85%, 96.48%, 99.26%, 97.19%.

Table 2. Classification accuracy for both 500 epochs and 4000 epochs

Person	Accuracy for 500 epochs	Accuracy for 4000 epochs
1	88%	95%
2	94%	98%
Average (%)	**91%**	**96.5%**

Test 3: The third test involves 3 models each of which have been trained on only one category: humans, animals or cars. The purpose of this test was to verify whether the type of image (content within the image) has a direct effect on the accuracy scores. Three identical systems were created, with the only difference being the testing and training image. For the first model we used pictures of humans (i.e. Donald trump and Barack Obama), the second model had pictures of animals (i.e. dog and cat) and the third model had pictures of cars (i.e. Lamborghini and Ferrari). Each training set contained 10 images, and the testing set contained 3 "unseen" images of the corresponding category from the aforementioned datasets. All images were of high quality and similar pixel dimensions, these variables were controlled to avoid picture quality, image quantity or the like influencing results.

Table 3 shows the results, having conducted the third test. Evidently the model trained on only human face images achieved the highest accuracy of 93%, compared to both car and animal images. Considering variables such as image quality/dimensionality and quantity having no effect on the results, this indicates the type of images does indeed influence accuracy results to an extent. The results although preliminary indicate the CNN system working best with human face images, thus the results from the first test may have been different in favor of the Caltech Face model had there been more training images.

Table 3. Classification accuracy for models trained on the human face, car and animal dataset

System	Human	Car	Animal
Average accuracy (%)	93	87	73

5 Discussion

The aim of this study was to find a model suitable for Transfer Learning, being able to achieve respectable accuracy scores within a short space of time and with limited computational power. The study addressed different aspects of Machine Learning and explained the principals behind the Convolutional Neural Network architecture. We were able to find a suitable architecture that allows image classification through Transfer Learning, this came in the form of Inception-v3. A series of tests were conducted to determine the usability of such a technique and whether it could be applied to different sets of data. In turn, we could prove the usefulness of Transfer Learning as the results from the tests proved retraining the Inception-v3 model on the CIFAR-10 dataset resulted in better results compared to that stated in the previous state-of-the-art works, whereby authors in [4] did not use Transfer Learning and instead used a CNN trained on the same dataset (CIFAR-10) from scratch.

The CIFAR-10 retrained model proposed within this paper achieved an overall accuracy of 70.1%, compared to the 38% achieved and stated in [4]. In addition, the proposed system had a 100% pass rate as every image tested was given the correct classification, though there was variation in accuracy/confidence scores. Furthermore, given the preliminary results obtained from the first two tests we could verify that number of epochs and quantity of images in a dataset had a direct stimulus on the accuracy achieved. That said, the quality of the images was also identified as a factor, given the Caltech Face dataset had far less images compared to the CIFAR-10 dataset, and it still managed to achieve reasonable results which were similar. This was due to the fact the images were higher in quality and had variety (i.e. different lighting conditions, expressions and angles) as opposed to the CIFAR-10 dataset, which had small generic images from basic angles, which proved to be useful in helping the model achieving respectable accuracy given the low quantity training set. The third test gave the notion that the type of image influences the accuracy of a pre-trained model. This can be useful in certain circumstances whereby determining a specific dataset is required. A dataset consisting of one or limited classes will prove more useful in terms of classification accuracy compared to a dataset consisting of several types.

The main limitations within this study were computational power and time. As we established in test 2, increasing the number of training steps (epochs) increased the classification accuracy, though also increasing the training time considerably. Given access to a GPU as opposed to a CPU would have resulted in better accuracy and time efficiency, though the results presented within this paper were sufficient in achieving our aims. Using more images for our training datasets would have resulted in better accuracy, though we were limited in that regard. Ideally, building a model from scratch would allow the customization of layers/weights giving better efficiency and a more accurate functionality. This option however was not required in this study as the aim was to compare against works that had already been completed. This study used a pre-trained model and with the use of Transfer Learning we were able to retrain the aforementioned models with a new dataset and in turn provide accurate image classification. The final layer of the pre-trained model was retrained to provide this classification, thus maintaining the key knowledge in terms of weights from the models

initial training and transferring that to the new dataset. The pretrained model (Inception-v3) used was sufficient in being able to provide reasonable classification accuracy given the low quantity training set, as it had initially been trained on a dataset consisting of a million images (ImageNet).

Given the findings of this paper, this provides a solid basis to advance the use of Transfer Learning in not only the model presented but other deep neural networks. The CNN model used can be refined and fine-tuned further by stacking additional layers and adjusting weights. There are numerous possibilities which can result in a more complex model able to achieve better accuracy results.

Future Work: As future work, having affirmed the notion of being able to use Transfer Learning on an existing model to achieve decent accuracy results in different domains permits many possibilities. This sort of application may be useful in class registration systems as well as being used as a biometric password. There is also the possibility to research different implementations of the system on a webserver, this would potentially allow users to use their face to verify their identity from any device and location in the world relating to the Internet of Things (IoT) as most systems/devices are in one way or another connected online or moving towards such a perception. There is still room to improve the accuracy results through implementation with the Inception-v4 model, increased epochs size and testing on larger datasets providing you have the time and resources. Another possibility would be combining a CNN with a Long-Short Term Memory (LSTM), this may be multifaceted, but in theory should help achieving better results and efficiency.

References

1. TensorFlow: Image Recognition (2018). https://www.tensorflow.org/tutorials/image_recognition. Accessed 20 Apr 2018
2. Christian Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovih, R.: Going Deeper with Convolutions (arXiv.org) (2015)
3. Lin, M., Chen, Q., Yan, S.: Network in Network. ICLR submission (arXiv.org) (2013)
4. Nguyen, N.: Computational Collective Intelligence. Springer, Cham (2016)
5. Arun, P., Katiyar, S.: A CNN based Hybrid approach towards automatic age registration. Geodesy Cartography **62**(1), 33–49 (2013)
6. Rohrer, B.: How do Convolutional Neural Networks work? (2016). http://brohrer.github.io/how_convolutional_neural_networks_work.html. Accessed 20 Apr 2018
7. ImageNet: About (2016). http://image-net.org/about-overview. Accessed 20 Apr 2018
8. Gao, Y., Mosalam, K.: Deep transfer learning for image-based structural damage recognition. Comput. Aided Civ. Infrastruct. Eng. (2018)
9. Larsen-Freeman, D.: Transfer of learning transformed. Lang. Learn. **63**, 107–129 (2013)
10. Krizhevsky, A.: The CIFAR-10 dataset (2009). https://www.cs.toronto.edu/~kriz/cifar.html. Accessed 20 Apr 2018
11. Vision Caltech: Computational Vision (2018). http://www.vision.caltech.edu/html-files/archive.html. Accessed 20 Apr 2018
12. Wu, J.: CNN for Dummies. Nanjing University (2015)

13. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE CVPR 2016: Computer Vision and Pattern Recognition (2016)
14. Mathworks.com: Convolutional Neural Network (2018). https://www.mathworks.com/content/mathworks/www/en/discovery/convolutional-neural-network.html. Accessed 20 Apr 2018