 doi:10.20944/preprints202210.0081.v1

Article

A model selection algorithm for complex multi-domain CNN systems based on feature-weights relation in deep learning

Eyad Alsaghir, Xiyu Shi* and Varuna De Silva

Institute for Digital Technologies, Loughborough University London, Queen Elizabeth Olympic Park, Here East, London E20 3BS, UK; e.alsaghir@lboro.ac.uk (EA); v.d.de-silva@lboro.ac.uk (VDS)
* Correspondence: x.shi@lboro.ac.uk; Tel.: +44-20-38051324

Abstract: Object recognition is an essential element of machine intelligence tasks. However, one model cannot practically be trained to identify all the possible objects it encounters. An ensemble of models may be needed to cater to a broader range of objects. Building a mathematical understanding of the relationship between various objects that share comparable outlined features is envisaged as an effective method of improving the model ensemble through a pre-processing stage, where these objects' features are grouped under a broader classification umbrella. This paper proposes a mechanism to train an ensemble of recognition models coupled with a model selection scheme to scale-up object recognition in a multi-model system. An algorithmic relationship between the learnt parameters of a trained classification model and the features of input images is presented in the paper for the system to learn the model selection scheme. The multiple models are built with a CNN structure, whereas the image features are extracted using a CNN/VGG16 architecture. Based on the models' excitation weights, a neural network model selection algorithm, which links a new object with the models and decides how close the features of the object are to the trained models for selecting a particular model for object recognition is developed and tested on a five-model neural network platform. The experiment results show the proposed model selection scheme is highly effective and accurate in selecting an appropriate model for a network of multiple models.

Keywords: CNN; AI; Causality; Understandability; Object Features; Excitation Weight; Multi-model Neural Network; Model Selection.

1. Introduction

Recent advances in neural networks have enabled many strenuous tasks to be accomplished by machines, sometimes surpassing human performance. Object recognition [1], Scene understanding [2], [3], image super-resolution [4] and image captioning [5] are a few of such machine intelligence tasks related to visual perception. In supervised learning with visual perception tasks, machines can learn from repeated measurements of studied phenomena and the associated frequency of different event outcomes. Such visual perception tasks are at the heart of Artificial Intelligence (AI). Different recognition techniques such as classification, time-series retrieval temporal features and spatial information verification require a measure of functional similarity between time series, temporal and/or spatial information included with the input data [6].

However, the human brain categorises things according to what the brain is looking for when it sees things [7]. That depends on what objects the brain sees and its imagery features during the brain analysis or focusing processes. For example, if humans recognise an image of the sea, they may expect straightaway to see ships, fish anglers, swimmers, and other water (sea)-related objects. Therefore, we can say that the model of the water-related objects will be selected and applied for further detection and analysis processes. This classification criterion flags a different approach in the classification processing

 doi:10.20944/preprints202210.0081.v1

mechanism by processing the recognised object features pre to the processing analysis procedure. However, a machine cannot easily and quickly build this recognition knowledge without human interference in defining the related model, especially if the machine is loaded with many different models. In some instances of problem analysing, our observation can extract a deficient level of features that forces us to know about the latent variables due to the usual difficulty in computing the probability densities [8]. This human recognition neural system inspired the development of the Artificial Neural Network (ANN) concept [9]. In 1943, Walter Pitts and Warren McCulloch created the first neural network computational model based on mathematical algorithms known as threshold logic [10]. These algorithms were the first computational description of multi-layered perceptron (MLP) neural network behaviour, which was the primary key in giving the machine the ability to generate recognition models for specified object classes. As a result of these models, and like the MLP, Convolutional Neural Network (CNN) has emerged in the development of ANN as a standard tool for computerized image classification [11], which has proved a decisive success in various computer vision tasks (e.g. video action recognition [12], image classification [13], etc.) The CNN is designed to deal with the available training data and various optimisation and activation functions to build an image recognition model by appropriately adapting its contents of convolution blocks with its learnable parameters, including the convolutional filter weights.

However, a significant concern with machine learning systems that deal with this technology field is the need for human interference in controlling machine learning [14]. The domain to be analysed in ANN systems must be pointed at with a human's help. This means the machine must be instructed on which model to use for a new test input image or perform a new training procedure of all added classes from scratch. Therefore, specifying a machine to deal with a particular domain is a considerable drawback in neural network technology. Hence, making the machine able to recognise the domain related to the test image according to its transformed features will enable the machine to pick the right domain, leading to a selection of the correct model and a more efficient learning procedure.

To address this gap, we investigate a possible link between a trained object recognition model's learnt weight vector and the extracted feature vector of a data sample. This link can be considered the fingerprint that proves the relation between the trained deep learning CNN model's parameters and the related image features from the input dataset used for training. By recognising this link, the system can use that knowledge as a model pre-selection scheme and immediately transfer the selected model's learnt knowledge to the newly recognised object for further development. However, this new selection scheme comes with severe adjustment and tuning challenges, from the massive development of AI model architectures to the various feature-extraction techniques.

This paper proposes a new neural network ensemble paradigm driven by an automatic model selection algorithm by investigating the hypothesized link relationship between the model's learnt parameters and the extracted features (i.e., the excitation weights in a neural network). This leads to exploring possible relations between the images' features used for training and the trained models' parameters. Hence, when applying to the yet-to-be-recognised image, the relationship can allow the machine to recognise the related model to continue with any nominated procedure without requesting human confirmation. This approach is instrumental in increasing the recognition level by widening the neural networks with rapid development of big data technology and high-performance computers.

In the paper, we developed an algorithmic linkage between the model's learnt parameters and the input images' extracted features. Using the link algorithm, we further developed a network model selection scheme for multi-domain neural networks and proved its effectiveness in model selection through a series of experiments. The proposed selection scheme would let the system save its computation power without trading off

 doi:10.20944/preprints202210.0081.v1

any future classification accuracy by efficiently generating a model ready for prediction without incurring extra computation resources on the actual training procedure.

The paper is arranged as follows: the related work is first discussed in Section 2, and the proposed linkage and network model selection algorithm are presented in Section 3. We describe the details of the network selection experiment and the results in Section 4. With the findings discussed in Section 5, we finally summarise the paper's conclusion in Section 6.

2. Related work

Humans identify unknown objects with their instinct to learn about them until they build corresponding knowledge at the end. This fact motivated Joseph et al. [15] to propose their Open World Object Detection (ORE) model as a novel solution to computer vision problems. The ORE model identified unknown objects and learned them incrementally without forgetting the previously learned classes. This provided a novel solution to the problem they formulated about open-world object detection based on energy-based unknown identification of the noticed separation between known and unknown classes for unknown detection. They used contrastive clustering that identifies how close a similar and dissimilar item can be for open-world learning. By differentiating the unknown instance feature representation from the other known ones and facilitating learning new class instances' feature representations without overlapping with the previous ones, the ORE reduced the incremental object detection setting confusion through characterising and identifying unknown instances. As a result, it is a model of incremental learning without forgetting.

Moreover, with the observation of the recognition system using CNN, whereas the CNN's filter dimensions are relatively fixed over time, a dramatic decrease in the number of excitation weights is noticed during technology development [16]. This observation reflects a practical link between the number of excitation weights and the model accuracy performance, especially when deep learning becomes the best tool that yields good accuracy in image classification with large input compared to traditional computer vision algorithms [17]. These findings encouraged Poojary and Pai [18] to perform a comparative study of the model's optimisation techniques in fine-tuned CNN models, concluding that fine-tuned transfer learning CNN models are best for building models with similar given tasks to the original task. However, they found that the performance of CNN models is heavily affected by the used model's optimisation techniques.

The basic CNN recognition computations are based on a multi-dimensional dot-product operation applied between the extracted feature vectors and the model's weights vectors [19]. According to the model's weights' sparsity, Hegde et al. [19] studied how weight repetition can be used during CNN inference to improve performance and save energy, and consequently, the Unique Weight CNN Accelerator (UCNN) was proposed. As a result, the generalisation of exploiting repetition in all zero weights improved the efficiency with UCNN, which approved the actual relation between the model performance and the input features for better computation processes up to 3.7 times on three trendy CNNs.

Other work in the literature studied the CNN models' excitation weights and their relation to recognising the tested images. In [20], the authors identified CNN's two significant weights characteristics, the magnitude and location, during training. They developed two separate weight excitation (WE) mechanisms with those useful characteristics. The WE mechanisms were applied to modify the backpropagation process via weights' reparameterisation. These two mechanisms improved accuracy on the various networks they tested, such as an increase of 1.3% to the previous accuracy rate of the ResNet50 models. The authors demonstrated the importance of weights in the convolution blocks in providing substantial performance gains without changing inference at the structure of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

CNN or adding computation cost. However, their work again needed human interference in choosing the studied domain and forwarding the decision to the system. Furthermore, a few studies have also worked on the relationship between the extracted features and the trained model's detection results [21], [22].

Han et al. [23] researched the intensity of computation processes and memory usage in neural networks to overcome deployment difficulties on embedded systems. They studied the relationship between the model's weights and the input training process to reduce the overall computation costs and overcome the fact that CNN froze the architecture during training. They proposed a three-step method of learning the essential connections by trimming the unimportant connections and retraining the network for weights fine-tuning. They decreased the number of weights by a factor of 9 – 13 times for various networks without suffering accuracy loss.

With the continuous development of deep learning, its applications in practical scenes have kept rising more than before. However, it became more challenging to train models with rare mission scenarios due to limited samples available for training. These difficulties raised the importance of model adaptation within the few-shot learning with limited supervised information related to the concerned task. Since fewer labelled samples are available with few-shot learning, it became necessary to apply features with fewer training samples as much as possible. Jiang et al. [24] proposed a new metric-based few-shot learning method called Multi-Scale Metric Learning (MSML). MSML uses a feature pyramid structure [25] and a pre-trained feature extraction network to create multi-scale feature maps, then accommodate a multi-scale different samples' feature comparison through learning a multi-scale relation creation network. They enhanced the multi-scale relation generation network metric-learning by proposing the inter-class and intra-class relation loss function (IIRL) to enlarge heterogeneous group discrimination samples and decrease the same class samples' distance. They applied the MSML method on miniImageNet and tieredImageNet and yielded superior few-shot learning classification results. However, it requires a comprehensive approach to target positioning, image segmentation, and recognition.

Finally, the proof of the existence of a linkage between the features extracted from a training image and the trained model's structure and excitation weights was explored by the mathematical relationship between the resulting model's parameters and the model's featured input, Alsaghir et al. [26]. In their work, A feature-weight (FW) linkage was built to calculate the primary model-related FW array for comparing the extracted features of a test image to allocate the related model correctly in a multi-model system. In applying such a linkage algorithm to a multi-domain machine learning system, the system can select the related model for the correct domain for further analysis without human interference.

3. The proposed method for network selection

For an object recognition system loaded with multiple models, it will be difficult for the system to decide which model to use for recognising the tested object without applying the new image to all models that generate a separate prediction for each model or retraining all different recognised classes by all models. To address this issue, the proposed network selection method uses a pre-processing stage to direct a new image to the related recognition model for further analysis without human interference. Building such a pre-processing algorithmic approach is useful for recognition systems that can mimic human reactions and starts their prediction process effectively straight away. It avoids further model training when detecting a new unknown object in a multi-model system by selecting the closely related model based on features extracted from the image.

The proposed method includes three main steps: 1) combining the trained and tested image features with the model's excitation weights in FW arrays ℓ_j and $\hat{\ell}_j$, which are

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

generated for each different CNN recognition model, j, and loaded into the system; 2) calculating the relative distances between arrays ℓ_j and $\hat{\ell}_j$, of the different CNN models by applying a sequence of deviation operations; and 3) comparing the distance between all possible network models to link the new unknown input image to an appropriately trained model. Figure 1 illustrates these processing steps.

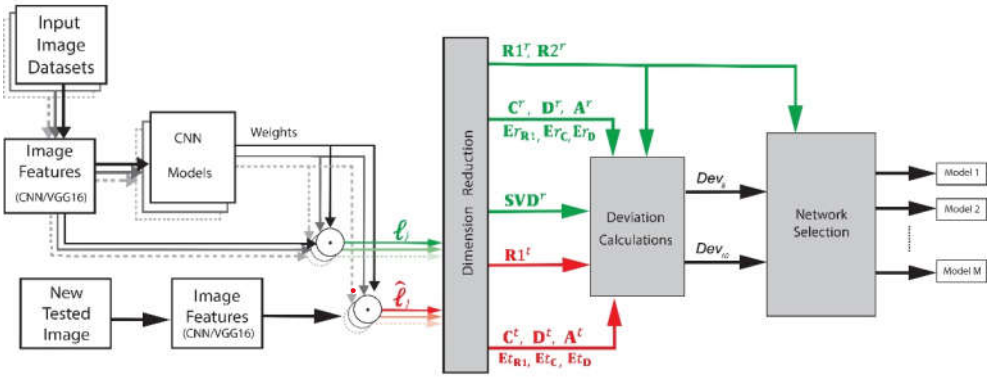


Figure 1. Block diagram of the main steps in the proposed network model selection algorithm. Each CNN model is built with three convolutional blocks. ℓ is the dot-product of trained image features and the model's excitation weights array, $\hat{\ell}$ is the dot-product of test image features and the model's excitation weights array, and Dev_8 and Dev_{10} are the deviation values between ℓ and $\hat{\ell}$. These values were presented to the Network Selection stage to analyse the relative distance between the feature-weight arrays, ℓ and $\hat{\ell}$, in accordance with the values of $\mathbb{R}1^r$ and $\mathbb{R}2^r$.

3.1. FW array generation

The FW linkage arrays are generated by applying a dot-product operation, denoted as $(\odot)$, between the extracted N features, $F = [f_1\ f_2\ \cdots\ f_N]^T$, of an object's image and the excitation weights, $W = [w_1\ w_2\ \cdots\ w_N]$, of the trained model, as shown in (1). Note that $[\cdot]^T$ is the transpose of an array, and $N = 25088$ due to CNN/VGG16 being used in the feature extraction [27].

$$F \odot W = (FW)^T = \begin{bmatrix} f_1 \cdot w_1 & f_2 \cdot w_1 & \cdots & f_N \cdot w_1 \\ f_1 \cdot w_2 & f_2 \cdot w_2 & \cdots & f_N \cdot w_2 \\ f_1 \cdot \vdots & f_2 \cdot \vdots & \ddots & f_N \cdot \vdots \\ f_1 \cdot w_N & f_2 \cdot w_N & \cdots & f_N \cdot w_N \end{bmatrix} \quad (1)$$

This array generation stage can be further divided into two steps, with each trained model loaded into the system. The first step links the original correctly predicted image's features with the related model's excitation weights. The second step links the extracted features of the unknown new test image with the same model's excitation weights.

As the first step, all images in the training datasets must go through the CNN/VGG16 feature extraction process, and the features are fed to the CNN model for training. A feature array, $\mathcal{F}_j^r$, of training image r with model j, can be constructed, such that $\mathcal{F}_j^r = [f_{j1}^r\ f_{j2}^r\ \cdots\ f_{jN}^r]^T$. An excitation weight array, $\mathcal{W}_j$, of each model j, is extracted from the model, such that $\mathcal{W}_j = \sum_i w_{j,b_i}$, where $w_{j,b_i} = [w_{j,b_i,1}\ w_{j,b_i,2}\ w_{j,b_i,N}]$ refers to the N excitation weights of the convolutional block b_i of model j, and $i \in [1,3]$ as there are three convolutional blocks in the proposed network models.

In accordance with (1), the link, ℓ_j^r, between model j and training image r is calculated as follows:

$$\ell_j^r = \mathcal{F}_j^r \odot \mathcal{W}_j = \sum_{i=1}^{3} \left(\mathcal{F}_j^r \odot w_{j,b_i}\right),\ \forall j \in [1, M], \quad (2)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

where ℓ_j^r is an $[N \times N]$ array and M is the number of models loaded into the system. In our model selection system in the experiment, M is 5. Note that hereafter $j \in [1, M]$ and $n \in [1, N]$ in all relevant descriptions.

A newly recognised object (image), t, from the test dataset also goes through the CNN/VGG16 for feature extraction in the second step. The test image t does not contribute to the model's training processes. Using the same method as described in the first step above, a feature array $\mathcal{F}_j^t$ for image t with model j can also be generated. Thus, similar to (2), a new link $\hat{\ell}_j^t$ between the model j and the features of test image t, can be built as follows:

$$\hat{\ell}_j^t = \mathcal{F}_j^t \odot \mathcal{W}_j = \sum_{i=1}^{3} (\mathcal{F}_j^r \odot w_{j,b_i}), \tag{3}$$

where $\hat{\ell}_j^t$ is also an $[N \times N]$ array.

3.2. Network selection algorithm

The network selection algorithm developed in this paper uses the linkage arrays, ℓ_j^r and $\hat{\ell}_j^t$ to estimate the relative distances in order to find the nearest test FW array to the expected model's FW arrays. It starts by reducing array dimensions and developing different deviation criteria to build a logical approach that defines the distances between the FW arrays. In the end, a distance comparison is performed to select the correctly related network model.

3.2.1. Dimension reduction

Since the FW arrays are size of [N×N] that has 629,407,744 elements, we need to reduce the array's dimensions to a level that can be tuned effectively. We aim to reduce the dimensions of these arrays to a much smaller size of $[9 \times 1]$ by extracting nine values from each FW array. This will allow a straightforward comparison between the new test images, old images used in training, and the trained recognition models. The values are extracted using the N^2 elements of the FW array, and extra $3N$ values that include the elements of a chosen row, a chosen column, and the array's diagonal elements to enhance the uniqueness.

We first generate a row-summation vector and a column-summation vector of the FW array for model j, as defined in (4.1) and (4.2):

$$A_{Rj} = \begin{bmatrix} \mathcal{E}_{R1,j} \\ \vdots \\ \mathcal{E}_{RN,j} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{N} \ell_j^r[1,k] \\ \vdots \\ \sum_{k=1}^{N} \ell_j^r[N,k] \end{bmatrix}, \tag{4.1}$$

$$A_{Cj} = \begin{bmatrix} \mathcal{E}_{C1,j} \\ \vdots \\ \mathcal{E}_{CN,j} \end{bmatrix} = \begin{bmatrix} \sum_{m=1}^{N} \ell_j^r[m,1] \\ \vdots \\ \sum_{m=1}^{N} \ell_j^r[m,N] \end{bmatrix}, \tag{4.2}$$

where A_{Rj} and A_{Cj} refer to vectors with elements $\mathcal{E}_{Rn,j}$ and $\mathcal{E}_{Cn,j}$ that are the summation of rows and columns of link array ℓ_j^r, $\forall n \in [1, N]$, respectively.

The minimum and maximum elements of A_{Rj} and A_{Cj}, denoted as $min\mathcal{E}_j$ and $max\mathcal{E}_j$, are found in (5.1) and (5.2). With each element being scaled to the range of [-1.0, 1.0] in accordance with (6), the vectors A_{Rj} and A_{Cj} are normalised and denoted as A'_{Rj} and A'_{Cj}, as shown in (7.1) and (7.2).

$$min\mathcal{E}_j = \min_{n \in [1,N]} (\mathcal{E}_{Rn,j}, \mathcal{E}_{Cn,j}) \tag{5.1}$$

$$max\mathcal{E}_j = \max_{n \in [1,N]} (\mathcal{E}_{Rn,j}, \mathcal{E}_{Cn,j}) \tag{5.2}$$

$$\mathcal{E}'_{n,j} = 2.0 * \left(\frac{\mathcal{E}_{n,j} - min\mathcal{E}_j}{max\mathcal{E}_j - min\mathcal{E}_j} \right) - 1.0 \tag{6}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

$$A'_{Rj} = [\mathcal{E}'_{R1,j}, \mathcal{E}'_{R2,j}, \cdots, \mathcal{E}'_{RN,j}]^T \quad (7.1)$$

$$A'_{Cj} = [\mathcal{E}'_{C1,j}, \mathcal{E}'_{C2,j}, \cdots, \mathcal{E}'_{CN,j}]^T \quad (7.2)$$

The means μ_{Rj}, μ_{Cj}, and the standard deviations σ_{Rj}, σ_{Cj}, of all elements in the normalised vectors A'_{Rj} and A'_{Cj}, are further calculated as in (8.1), (8.2), (9.1) and (9.2), respectively.

$$\mu_{\mathrm{R}j} = \left(\sum_{n=1}^{N} \mathcal{E}'_{Rn,j}\right)/N \quad (8.1)$$

$$\mu_{Cj} = \left(\sum_{n=1}^{N} \mathcal{E}'_{Cn,j}\right)/N \quad (8.2)$$

$$\sigma_{Rj} = \sqrt{\sum_{n=1}^{N} (\mathcal{E}'_{Rn,j} - \mu_{\mathrm{R}j})^2/N} \quad (9.1)$$

$$\sigma_{Cj} = \sqrt{\sum_{n=1}^{N} (\mathcal{E}'_{Cn,j} - \mu_{\mathrm{C}j})^2/N} \quad (9.2)$$

We further define two new parameters, $\delta_{Rn,j}$ and $\delta_{Cn,j}$, as element distances to the means for vectors A'_{Rj} and A'_{Cj}, respectively, as follows:

$$\delta_{Rn,j} = \left(\mathcal{E}'_{Rn,j} - \mu_{\mathrm{R}j}\right)/\sigma_{Rj} \quad (10.1)$$

$$\delta_{Cn,j} = \left(\mathcal{E}'_{Cn,j} - \mu_{\mathrm{C}j}\right)/\sigma_{Cj} \quad (10.2)$$

According to (11.1) and (11.2), the row $\hbar$ and column d with the maximum element distances will be used for network selection with the FW link arrays ℓ_j^r and $\hat{\ell}_j^t$ of network model j.

$$\hbar = \arg\left(\max_{n\in[1,N]}\left(\delta_{Rn,j}\right)\right) \quad (11.1)$$

$$d = \arg\left(\max_{n\in[1,N]}\left(\delta_{Cn,j}\right)\right) \quad (11.2)$$

Based on $\hbar$, d, and FW arrays ℓ_j^r and $\hat{\ell}_j^t$, nine other values are obtained for network model selection, as follows:

1. A pair of sums of elements in row $\hbar$ for the training image r and test image t of model j, respectively, as shown in (12.1) and (12.2).

$$\mathbb{R}1_j^r = \sum_{k=1}^{N} \ell_j^r[\hbar, k] \quad (12.1)$$

$$\mathbb{R}1_j^t = \sum_{k=1}^{N} \hat{\ell}_j^t[\hbar, k] \quad (12.2)$$

2. A sum of next row elements to $\hbar$, $\mathbb{R}2_j^r$, for the training images$r$ of model j, according to (12.3).

$$\mathbb{R}2_j^r = \sum_{k=1}^{N} \ell_j^r[\hbar + 1, k] \quad (12.3)$$

3. A pair of sums of column d elements in FW arrays, denoted as $\mathbb{C}_j^r$ and $\mathbb{C}_j^t$, for the training image r and test image t of model j, respectively, according to (13.1) and (13.2).

$$\mathbb{C}_j^r = \sum_{n=1}^{N} \ell_j^r[n, d] \quad (13.1)$$

$$\mathbb{C}_j^t = \sum_{n=1}^{N} \hat{\ell}_j^t[n, d] \quad (13.2)$$

4. A pair of sums of diagonal elements in FW arrays, denoted as $\mathbb{D}_j^r$ and $\mathbb{D}_j^t$, for the training images r and test image t of model j, according to (14.1) and (14.2).

$$\mathbb{D}_j^t = \sum_{k=1}^{N} \hat{\ell}_j^t[k, k] \quad (14.1)$$

$$\mathbb{D}_j^r = \sum_{k=1}^{N} \ell_j^r[k, k] \quad (14.2)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

5. A pair of sums of all elements in FW arrays, denoted as $\mathbb{A}_j^t$ and $\mathbb{A}_j^r$, for the test image t and training images r of model j, according to (15.1) and (15.2).

$$\mathbb{A}_j^t = \sum_{m=1}^{N} \sum_{k=1}^{N} \hat{\ell}_j^t\,[m,k] \tag{15.1}$$

$$\mathbb{A}_j^r = \sum_{m=1}^{N} \sum_{k=1}^{N} \ell_j^r\,[m,k] \tag{15.2}$$

6. A pair of ratios of row $\hbar$ sums to the sums of all elements in FW arrays, denoted as $\mathbb{E}_{\mathbb{R}1,j}^t$ and $\mathbb{E}_{\mathbb{R}1,j}^r$, for the test image t and training images r of model j, according to (16.1) and (16.2).

$$\mathbb{E}_{\mathbb{R}1,j}^t = \frac{\mathbb{R}1_j^t}{\mathbb{A}_j^t} \tag{16.1}$$

$$\mathbb{E}_{\mathbb{R}1,j}^r = \frac{\mathbb{R}1_j^r}{\mathbb{A}_j^r} \tag{16.2}$$

7. A pair of ratios of column $\mathcal{d}$ sums to the sums of all elements in FW arrays, denoted as $\mathbb{E}_{\mathbb{C},j}^t$ and $\mathbb{E}_{\mathbb{C},j}^r$, for the test image t and training images r of model j, according to (17.1) and (17.2).

$$\mathbb{E}_{\mathbb{C},j}^t = \frac{\mathbb{C}_j^t}{\mathbb{A}_j^t} \tag{17.1}$$

$$\mathbb{E}_{\mathbb{C},j}^r = \frac{\mathbb{C}_j^r}{\mathbb{A}_j^r} \tag{17.2}$$

8. A pair of ratios of diagonal sums to the sums of all elements in FW arrays, denoted as $\mathbb{E}_{\mathbb{D},j}^t$ and $\mathbb{E}_{\mathbb{D},j}^r$, for the test image t and training images r of model j, according to (18.1) and (18.2).

$$\mathbb{E}_{\mathbb{D},j}^t = \frac{\mathbb{D}_j^t}{\mathbb{A}_j^t} \tag{18.1}$$

$$\mathbb{E}_{\mathbb{D},j}^r = \frac{\mathbb{D}_j^r}{\mathbb{A}_j^r} \tag{18.2}$$

9. Extracting a singular value $\mathbb{SVD}_j^r$ with the Singular Value Decomposition (SVD), from the FW linkage array, ℓ_j^r, for training image r and model j [28]. The array ℓ_j^r can be factorized as in (19):

$$\ell_j^r = UZV^T, \tag{19}$$

where U is a $[N \times N]$ matrix of the orthonormal eigenvectors of $(\ell_j^r \times \ell_j^{r^T})$, Z is a $[N \times N]$ diagonal matrix of the singular values, which are the square roots of the eigenvalues of $(\ell_j^{r^T} \times \ell_j^r)$, and V^T is the transpose of a $[N \times N]$ matrix containing the orthonormal eigenvectors of $(\ell_j^{r^T} \times \ell_j^r)$. Through solving $Z - (\mathbb{SVD}_j^r \times I) = 0$, where I is the identity matrix, a suitable singular value $\mathbb{SVD}_j^r$ can be found for ℓ_j^r.

3.2.2. Deviation values calculations

The proposed model selection system generates deviation values from the previously generated nine values in (12.1-19). These deviation values indicate the relative difference of the generated nine values between the FW arrays in training and test, and are the primary tool in making the comparison mathematically between the FW arrays. It starts by creating different criteria of deviation comparison between each model's FW array in training, ℓ_j^r, and the test image-related FW arrays, $\hat{\ell}_j^t$, with mathematical connections becoming available for model selection during the comparison.

In order to generate the deviation values, we first calculate a group of imagined phases between the extracted pair of ratios of $\hat{\ell}_j^t$ and ℓ_j^r, including between each model's

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

pair of row-values of $\mathbb{R}1_j^r$ and $\mathbb{R}1_j^t$, pair of column-values of $\mathbb{C}_j^r$ and $\mathbb{C}_j^t$, pair of diagonal-values of $\mathbb{D}_j^r$ and $\mathbb{D}_j^t$, pair of the array's all-element values of $\mathbb{A}_j^r$ and $\mathbb{A}_j^t$, pair of the array's row $\hbar$ ratio values of $\mathbb{E}_{\mathbb{R}1,j}^r$ and $\mathbb{E}_{\mathbb{R}1,j}^t$, pair of array's column d values of $\mathbb{E}_{\mathbb{C},j}^r$ and $\mathbb{E}_{\mathbb{C},j}^t$, and pair of array's diagonal ratio values of $\mathbb{E}_{\mathbb{D},j}^r$ and $\mathbb{E}_{\mathbb{D},j}^t$, related to each model j. These imagined phases, $\Phi_{i,j}, \forall i \in [1,7]$, are calculated as in (20.1-20.7). An average phase, Θ_j, for each model j is obtained in (21).

$$\Phi_{1,j} = arctan\left(\frac{\mathbb{R}1_j^r}{\mathbb{R}1_j^t}\right) \tag{20.1}$$

$$\Phi_{2,j} = arctan\left(\frac{\mathbb{C}_j^r}{\mathbb{C}_j^t}\right) \tag{20.2}$$

$$\Phi_{3,j} = arctan\left(\frac{\mathbb{D}_j^r}{\mathbb{D}_j^t}\right) \tag{20.3}$$

$$\Phi_{4,j} = arctan\left(\frac{\mathbb{A}_j^r}{\mathbb{A}_j^t}\right) \tag{20.4}$$

$$\Phi_{5,\mathrm{j}} = \arctan\left(\frac{\mathbb{E}_{\mathbb{R}1,\mathrm{j}}^{\mathrm{r}}}{\mathbb{E}_{\mathbb{R}1,\mathrm{j}}^{\mathrm{t}}}\right) \tag{20.5}$$

$$\Phi_{6,j} = arctan\left(\frac{\mathbb{E}_{\mathbb{C},j}^r}{\mathbb{E}_{\mathbb{C},j}^t}\right) \tag{20.6}$$

$$\Phi_{7,j} = arctan\left(\frac{\mathbb{E}_{\mathbb{D},j}^r}{\mathbb{E}_{\mathbb{D},j}^t}\right) \tag{20.7}$$

$$\Theta_j = \left(\sum_{i=1}^{7} \Phi_{i,j}\right)/7 \tag{21}$$

These calculated average phases, Θ_j, for FW array of training image r and test image t of each model j, are used to calculate the first relative deviation distance, denoted as $Dev_{1,j}$, as defined in (22).

$$Dev_{1,j} = \prod_{\substack{n=1 \\ n \neq j}}^{M} (\Theta_j - \Theta_n) \tag{22}$$

The system gets the second relative distance value, $Dev_{2,j}$, by applying a non-linear function to the average phase Θ_j and scaled with the singular value $\mathbb{SVD}_j^r$ for each model j. Here we use $arctan$ as the non-linear function, $Dev_{2,j}$ is defined as in (23).

$$Dev_{2,j} = \arctan(\Theta_j) \times \mathbb{SVD}_j^r \tag{23}$$

The third deviation value, $Dev_{3,j}$, is generated by calculating the difference between the pair of ratios of $\mathbb{E}_{\mathbb{R}1,j}^t$, and $\mathbb{E}_{\mathbb{R}1,j}^r$ from the $\hat{\ell}_j^t$ and ℓ_j^r arrays, respectively, as shown in (24).

$$Dev_{3,j} = \mathbb{E}_{\mathbb{R}1,j}^t - \mathbb{E}_{\mathbb{R}1,j}^r \tag{24}$$

The fourth deviation value, $Dev_{4,j}$, is defined as in (25).

$$Dev_{4,j} = \frac{[Dev_{3,j} - Dev_{2,j}]}{Dev_{3,j}} \tag{25}$$

We then construct two unique arrays, $\mathbb{S}^t$ and $\mathbb{S}^r$, as shown in (26.1) and (26.2), and the determinant values for each of $\mathbb{S}_j^t$ and $\mathbb{S}_j^r$, as shown in (27.1) and (27.2). The fifth deviation value, $Dev_{5,j}$, of model j is then defined as in (28).

$$\mathbb{S}_j^t = \begin{bmatrix} \mathbb{R}1_j^t & \mathbb{C}_j^t \\ \mathbb{D}_j^t & \mathbb{A}_j^t \end{bmatrix} \tag{26.1}$$

$$\mathbb{S}_j^r = \begin{bmatrix} \mathbb{R}1_j^r & \mathbb{C}_j^r \\ \mathbb{D}_j^r & \mathbb{A}_j^r \end{bmatrix} \tag{26.2}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

$$Det_j^t = \left[\mathbb{R}1_j^t \times \mathbb{A}_j^t\right] - \left[\mathbb{C}_j^t \times \mathbb{D}_j^t\right] \quad (27.1)$$

$$Det_j^r = f_{Det}(\mathbb{S}_j^r) = \left[\mathbb{R}1_j^r \times \mathbb{A}_j^r\right] - \left[\mathbb{C}_j^r \times \mathbb{D}_j^r\right] \quad (27.2)$$

$$Dev_{5,j} = Det_j^t - Det_j^r \quad (28)$$

The sixth deviation value, $Dev_{6,j}$, of model j, is defined in (29).

$$Dev_{6,j} = arctan[(tan\,(Dev_{5,j} \times Dev_{4,j}/\mathbb{C}_j^t - \mathbb{C}_j^r)) + tan(Dev_{4,j})] \quad (29)$$

The system defines the seventh deviation value, $Dev_{7,j}$, of model j, as in (30), which links the sixth and first deviation values.

$$Dev_{7,j} = Dev_{6,j} \times Dev_{1,j} \quad (30)$$

Afterwards, the system compares the last generated deviation value, $Dev_{7,j}$, with the same values of all tested models j for selected test image t according to (31).

$$Dev_{8,j} = \frac{\prod_{\substack{n=1 \\ n\neq j}}^{M}(|Dev_{7,j} - Dev_{7,n}|)}{Dev_{5,j}} \times Dev_{1,j} \quad (31)$$

Furthermore, the system calculates another value, $Dev_{9,j}$, as defined in (32).

$$Dev_{9,j} = \prod_{\substack{n=1 \\ n\neq j}}^{M}\left(\left|Dev_{1,j} - Dev_{1,n}\right|\right) \quad (32)$$

Finally, the system uses the last deviation value, $Dev_{10,j}$ to link $Dev_{8,j}$ and $Dev_{9,j}$ with the ratio values $\mathbb{E}_{\mathbb{D},j}^t$ and $\mathbb{E}_{\mathbb{C},j}^t$, and $\mathbb{R}2_j^r$, as defined in (33).

$$Dev_{10,j} = (Dev_{9,j}/Dev_{8,j})/\mathbb{E}_{\mathbb{D},j}^t \times \mathbb{E}_{\mathbb{C},j}^t \times \mathbb{R}2_j^r \quad (33)$$

3.2.3. Network selection algorithm

The network selection algorithm, as shown in (34), compares the relative distances between the FW arrays, $\hat{\ell}_j^t$ and ℓ_j^r, according to the specially developed deviation criteria in (33) of each model j, to select the related network model with the shortest distance.

We define the rules for choosing the related network model as: the selected model, $\bar{\jmath}$, is the one with the minimum or maximum $Dev_{10,j}$, depending on the signs of $\mathbb{R}1_j^r$, $\mathbb{R}2_j^r$, $Dev_{8,j}$, and $Dev_{10,j}$, as shown in (34).

$$\bar{\jmath} = \begin{cases} \arg\left(\min_{j\in[1,M]}\left(Dev_{10,j}\right)\right), & if\ \mathbb{R}1_j^r > \mathbb{R}2_j^r, Dev_{8,j} > 0, (\mathbb{R}2_j^r \times Dev_{10,j}) > 0 \\ \arg\left(\max_{j\in[1,M]}\left(Dev_{10,j}\right)\right), & if\ \mathbb{R}1_j^r < \mathbb{R}2_j^r, Dev_{8,j} < 0, (\mathbb{R}2_j^r \times Dev_{10,j}) < 0 \\ 0, Otherwise \end{cases}, \quad (34)$$

where $\bar{\jmath} = 0$ represents a failed network selection.

The overall selection algorithm can be described as follows:

Algorithm: Network Selection

1. *procedure* Array_Generation WeightsFeatures($\boldsymbol{w}_{i,j}, \boldsymbol{\mathcal{F}}_{i,j}$)
   ▷ **generate object's image features**
2. *if* **Image == trained Image**
3. $\boldsymbol{\mathcal{F}} \leftarrow \boldsymbol{\mathcal{F}}_{i,j}^r$
4. *else*
5. $\boldsymbol{\mathcal{F}} \leftarrow \boldsymbol{\mathcal{F}}_{i,j}^t$
   ▷ **generate trained model weights**
6. $\boldsymbol{w} \leftarrow \boldsymbol{w}_{i,j}$
   ▷ **generate features$_{weights}$array**
7. $\boldsymbol{\ell} = \boldsymbol{w} \,.\, \boldsymbol{\mathcal{F}}$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

procedure Dimension_Reduction ReducedValues(ℓ_j^r), ($\hat{\ell}_j^t$)
▷ **generate array related reduced values**
  Select a particular row and column sequence ($\hbar$), (d).
  for $k = 1 \rightarrow N$
    $\mathbb{R}1 \leftarrow \mathbb{R}1 + \ell[\hbar, k]$
    $\mathbb{R}2 \leftarrow \mathbb{R}2 + \ell[\hbar + 1, k]$
    $\mathbb{C} \leftarrow \mathbb{C} + \ell[m, d]$
    $\mathbb{D} \leftarrow \mathbb{D} + \ell[k, k]$
  for $m = 1 \rightarrow N$
    for $k = 1 \rightarrow N$
      $\mathbb{A}1 \leftarrow \mathbb{A}1 + \ell[m, k]$
    end
    $\mathbb{A} \leftarrow \mathbb{A} + \mathbb{A}1$
  end
  $\mathbb{E}_{\mathbb{R}1} \leftarrow \mathbb{R}1/\mathbb{A}$
  $\mathbb{E}_{\mathbb{C}} \leftarrow \mathbb{C}/\mathbb{A}$
  $\mathbb{E}_{\mathbb{D}} \leftarrow \mathbb{D}/\mathbb{A}$
  $\mathbb{SVD} \leftarrow \mathbb{SVD}$

procedure Deviation_Values DevValues($\mathbb{R}1, \mathbb{C}, \mathbb{D}, \mathbb{A}, \mathbb{SVD}$)
▷ **generate first, second, third, fourth deviation values**
  for $j = 1 \rightarrow M$
    Building $\Phi_{1,j}$ *to* $\Phi_{7,j}$ *with (20.1-20.7)*
    Building phase average Θ_j *with (21)*
    Building $Dev_{1,j}$, $Dev_{2,j}$, $Dev_{3,j}$, $Dev_{4,j}$ *with (22-25)*
  end
▷ **generate fifth, sixth, seventh, eight, ninth, tenth deviation value**
  for $j = 1 \rightarrow M$
    Building $(\mathbb{S}^t)_j, (\mathbb{S}^r)_j$ *array with (26.1) and (26.2)*
    Generate $(Det^t)_j, (Det^r)_j$ *determinant, with (27.1) and (27.2)*
    Building the fifth, sixth, seventh, eighth, ninth, and tenth deviation values with (28), (29), (30), (31), (32), and (33).
  end

procedure Model_Selection Model $S\left((Dev_{10})_j \middle| {}_{(\mathbb{R}1^r)_j,(\mathbb{R}2^r)_j,(Dev_8)_j,(Dev_{10})_j}\right)$
▷ **model selection**
  for $j, n = 1 \rightarrow M$
    if $(\mathbb{R}1^r)_j > (\mathbb{R}2^r)_j$,
      if $(Dev_8)_j > 0$,
        if $[(\mathbb{R}2^r)_j \times (Dev_{10})_{j]} > 0]$
          Select model j ***with*** $\arg\left(\min_{j\in[1,M]}(Dev_{10,j})\right)$
        else
          Select model $j = 0$
      end
      elseif $(\mathbb{R}1^r)_j < (\mathbb{R}2^r)_j$
        if $(Dev_8)_j < 0$,
          if $(\mathbb{R}2^r)_j \times (Dev_{10})_j < 0$
            Select model j ***with*** $\arg\left(\max_{j\in[1,M]}(Dev_{10,j})\right)$
        else
          Select model $j = 0$
      End
      return j
  end

4. Result discussion

4.1. Experiment setup

To evaluate the proposed network selection algorithm, we built a neural network system with five different neural network models, as listed in Table 1. The first model was adapted to distinguish between cats and dogs from a dataset extracted from the Dogs vs Cats dataset of 25000+ images available from the Kaggle [29]. The second model was trained to distinguish three different fruit types (Apple Red Yellow 2, Cantaloupe 1, and Orange) within the Fruits_360-Kaggle dataset [30]. The third model is to recognise seven types of facial emotions using the AFEW2.0 dataset [31], [32]. The fourth model is to classify four classes of objects, including aeroplanes, cars, flowers, and motorbikes extracted from the Natural Images-Kaggle dataset of 6899 images [33]. Finally, the fifth model is built to classify six classes of objects, including bathrooms, closets, computer rooms,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

garages, hospital rooms, and libraries extracted from the Indoor_Images-Kaggle dataset generated for object recognition [34]. The algorithm is implemented using Python (version 3.8.5) and TensorFlow (version 2.3.1) library.

Each model in the system is built with three blocks in series. There are two convolutional layers with ReLU activation functions, a Batch Normalisation layer after each convolutional layer, a Max-Pooling layer and a Dropout layer in each of the first two blocks. The third block is configured with one convolutional layer with ReLU activation function, followed by a Batch Normalisation layer and a Max-Pooling layer flattened into two dense Fully Connected layers along with ReLU and Softmax functions to indicating the results of object classification, as shown in Figure. 2. The details of the overall system configuration are listed in Table 1.

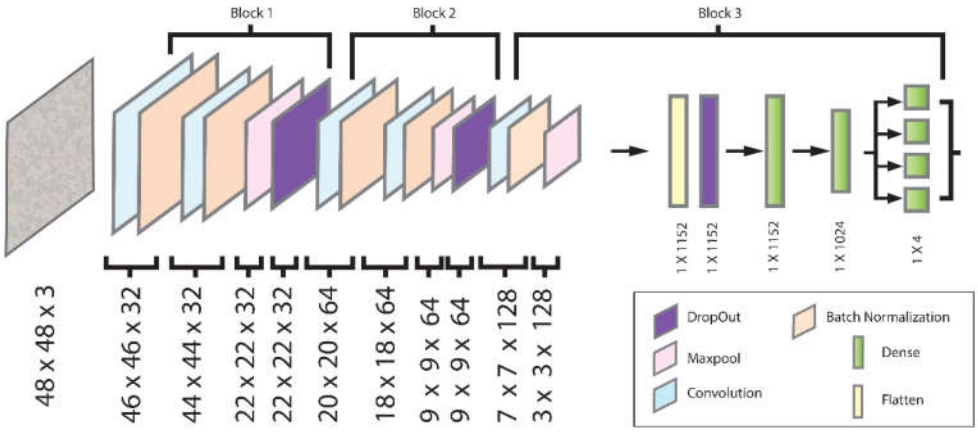


Figure 2. Illustration of a CNN model used to build the five-model neural network selection system. This model is designed to classify four objects (aeroplanes, cars, flowers, and motorbikes). Other models in the system have a similar structure but are trained for different classification tasks.

Table 1. Configuration details of a CNN model in the five-model neural network selection system. This model is designed for the classification of four objects.

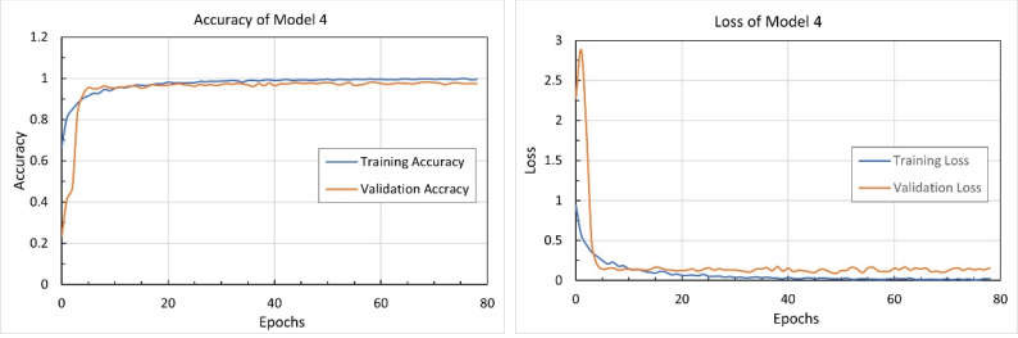
Block	Layer No.	Layer Type	Input Data Dimension	Output Data Dimension	Number of Parameters
	1	Input	3 × 48 × 48	3 × 48 × 48	
First Block	2	Conv2D (activation = relu)	3 × 48 × 48	32 × 46 × 46	896
	3	Batch Normalization (axis=-1)	32 × 46 × 46	32 × 46 × 46	128
	4	Conv2D (activation = relu)	32 × 46 × 46	32 × 44 × 44	9248
	5	Batch Normalization (axis=-1)	32 × 44 × 44	32 × 44 × 44	128
	6	MaxPooling2D (pool_size = (2, 2))	32 × 44 × 44	32 × 22 × 22	0
	7	Dropout (p-rate = 0.25)	32 × 22 × 22	32 × 22 × 22	0
	8	Conv2D	32 × 22 × 22	64 × 20 × 20	18496

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

		(activation = relu)			
Second Block	9	Batch Normalization (axis = -1)	64 × 20 × 20	64 × 20 × 20	256
	10	Conv2D (activation=relu)	64 × 20 × 20	64 × 18 × 18	36928
	11	Batch Normalization (axis = -1)	64 × 18 × 18	64 × 18 × 18	256
	12	MaxPooling2D (pool_size = (2, 2))	64 × 18 × 18	64 × 9 × 9	0
	13	Dropout (p-rate = 0.25)	64 × 9 × 9	64 × 9 × 9	0
Third Block	14	Conv2D (activation = relu)	64 × 9 × 9	128 × 7 × 7	73856
	15	Batch Normalization (axis = -1)	128 × 7 × 7	128 × 7 × 7	512
	16	MaxPooling2D (pool_size = (2, 2))	128 × 7 × 7	128 × 3 × 3	0
	17	Flatten	128 × 3 × 3	1152	0
	18	Dropout (p-rate = 0.5)	1152	1152	0
	19	Dropout (p-rate = 0.5)	1152	1024	1180672
	20	Dense (activation = softmax)	1024	4	2050

4.2. Performance in training

The network selection system is composed of five different models. Each model is trained for different classification tasks in 80 epochs with the selected dataset and other experiment setups described in the previous section. For example, Figure 3 shows the performance of Model 4 in terms of accuracy and loss in training and validation. This model is trained to classify four distinctive classes with the Natural Images-Kaggle dataset from [33]. We can see, from Figure 3, that Model 4 is converging quickly in training and is stable after about 40 epochs. The best training and validation accuracy are 99.8% and 98.1%, respectively.



a) b)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

Figure **3.** Training and validation performance of a model in the five-model neural network selection system. a) Accuracy of model 4, and b) loss of model 4.

4.3. Performance of network selection algorithm

4.3.1 Evaluation criterion

The performance evaluation is based on the accuracy of model selection in the system. We define correct selections as the sum of true-positive (TP) selections and the true-negative (TN) selections, incorrect selections as the sum of false-positive (FP) selections and false-negative (FN) selections. The selection accuracy is the ratio of correct selections to the sum of correct and incorrect selections, as defined in (35).

$$Accuracy = \frac{Correct\ Selections}{Correct\ Selections + Incorrect\ Selections} = \frac{TP+TN}{TP+TN+FP+FN} \tag{35}$$

We aim for a high level of accuracy of model selection for the proposed algorithm. However, high accuracy can sometimes be misleading. For example, in testing the selection of Model 1, we have 200 images from the dataset of Model 1 and 1000 images from other models' datasets. If the proposed selection algorithm is built in a way that it always selects models other than Model 1 for all images, it will achieve an accuracy of 1000/1200 = 83.33%. This is highly misleading as our algorithm is unable to select the correct model. Therefore, the selection accuracy alone is not able to determine if the selection algorithm is good or bad, but accuracy combined with precision, recall, and F1_Score can give us a good indication of the performance of the algorithm. For this reason, we also use selection precision, recall, and F1_Score in evaluating the performance of the proposed algorithm.

Precision is the ratio of the true positive selections to the total number of positive selections, as shown in (36). Recall calculates the ratio of true positive selections to the total number of positive labels, including true positive and false negative selections, as defined in (37). The F1-Score is the harmonic mean of precision and recall, as expressed in (38). A high F1-Score means a high value for both recall and precision, with a score of 1.0 representing the perfect precision and recall of the algorithm.

$$Precision = \frac{Ture\ Positive\ Selections}{True\ Positive\ Selections+False\ Positive\ Selections} = \frac{TP}{TP+FP} \tag{36}$$

$$Recall = \frac{True\ Positive\ Selections}{Total\ Number\ of Positie\ Labels} = \frac{TP}{TP+FN} \tag{37}$$

$$F1\_Score = 2.0 \times \frac{Recall \times Precision}{Recall+Precision} \tag{38}$$

4.3.2 Result analysis

To evaluate the performance of the network selection algorithm with the experimental five-model system, a total of 1200 images were randomly selected for testing. These images are from five different datasets that belong to the five models, with 200 images for each model except for Model 3, where 400 images are used. All images in the experiment have not been used in the training and validation of the system. After applying the network selection algorithm in (34), we examine whether the correct model is selected or not for the images for further processing by the system. The details of the experiment results are listed in Table 2.

Table 2. Results of model selection accuracy of the proposed algorithm on the multi-model system

Models Tested	Number of Images Tested	Models Selected					Selection Accuracy
		Model 1	Model 2	Model 3	Model 4	Model 5	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

Model 1	200	**172**	4	7	5	12	0.9633
Model 2	200	1	**183**	4	1	11	**0.9767**
Model 3	400	6	0	**382**	3	9	0.9692
Model 4	200	7	3	2	**181**	7	**0.9767**
Model 5	200	2	4	6	0	**188**	0.9575
Average							*0.9687*

Out of a total number of 1200 test images, 200 images are from the dataset of Model 1. The algorithm correctly selects 172 images for Model 1 and incorrectly allocates 28 images to other models. For the remaining 1000 images that are not from the dataset of Model 1, the algorithm incorrectly selects 16 of them for Model 1 and correctly classifies 984 images as not for Model 1. This gives a TP=127, FN=28, FP=16, and TN=984 for Model 1, representing a selection accuracy of 96.33% by applying (35). Similarly, we can get the selection accuracies for other models, as listed in the last column in Table 2. Other evaluation metrics – precision, recall and F1_Score, along with the accuracy of the network selection algorithm for each of the models, are also calculated according to (36-38) and listed in Table 3.

Table 3. Results of model selection precision, recall and F1_Score of the proposed algorithm on the multi-model system

Models Tested	Metrics of Network Selection			
	Accuracy	Precision	Recall	F1_Score
Model 1	0.9633	0.9149	0.8600	0.8866
Model 2	**0.9767**	0.9433	0.9150	0.9289
Model 3	0.9692	**0.9526**	**0.9550**	**0.9538**
Model 4	**0.9767**	**0.9526**	0.9050	0.9282
Model 5	0.9575	0.8282	0.9400	0.8806
Average	0.9687	0.9183	0.9150	0.9156

It is noted, from Table 2, that the best selection accuracy achieved is 97.67% for Model 2 and 4, the lowest model selection accuracy is 95.75.0% for Model 5, and the average accuracy of model selection of the algorithm is 96.87% in the experiment. Although the algorithm selects the greatest number of images from dataset of Model 5 correctly, it also has 39 images, the highest number among all models, that are from other models' datasets is allocated to Model 5 incorrectly. This can be translated into Model 5 having the lowest number of TN selections. Therefore, the selection accuracy for Model 5 is the lowest among the five models in the experiment system.

Amongst the five different models in the experiment, Table 3 shows that the selection algorithm has the lowest recall value of 86% for Model 1 and the lowest precision value of 82.82% for Model 5. It is obvious, in Table 2, that Model 1 has the largest number of FN selections (i.e., any selections other than Model 1) as indicated in the row of Model 1, while the column of Model 5 shows that Model 5 has the greatest number of FP selections (i.e., incorrectly selected Model 5 for images of other models). Therefore, Model 5 has the least precision value and Model 1 has the least recall value according to (36) and (37), respectively. Comparing with the F1_Scores of Models 1 and 5, there is little difference between them; hence the proposed selection algorithm performs the same for both models.

We also note, in Table 3, that the proposed selection algorithm has similar performance metrics for both Models 2 and 4, which are better than that of Models 1 and 5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 October 2022 doi:10.20944/preprints202210.0081.v1

Among all models tested in the experiment, the selection algorithm performed best for Model 3 as three out of the model's four metrics are the best in comparison with metrics of other models. The lowest ratios of false negative selections and false positive selections for Model 3 contribute to the algorithm performing best in this case.

With an average selection accuracy of 96.87% and more than 91% of average metrics of precision, recall and F1_Score, the FW-based network selection algorithm is perceived effective in choosing an appropriate model for images for further processing in a multiple complex neural network system on the benchmark datasets.

4.4 Limitation

According to (11.1) and (11.2), the row h and column d with the maximum element distances in the FW arrays are selected in building the parameters of the network selection algorithm. In order to understand the effectiveness of this row and column selection method, we have also used randomly selected FW rows and columns in the proposed network selection algorithm. The selection accuracies with randomly selected FW rows and columns varying between 72~89% are achieved. This is not significantly different with a completely failed design of the selection algorithm, which could have a theoretical selection accuracy of 83.33% in the experiment. Although the proposed network selection algorithm achieves a relatively high selection accuracy by using the maximum element distances in selecting the FW rows and columns, other methods to select suitable representative parameters of the FW arrays could be deployed in the algorithm and tested in the experiment. There is great potential for further optimisation and improvement to attain higher performance metrics of accuracy, precision, and recall.

One application of the proposed algorithm is to choose an appropriate neural network model for the input image for further processing in a complex multi-model system, hence, to improve the efficiency and effectiveness of the system. However, if the algorithm selects an incorrect network model for processing the input image further, the output of the system will be completely wrong, and the consequence could be catastrophic. This may happen as our algorithm does not select models with 100% accuracy. Therefore, it is vital to achieve a very high selection accuracy in such a multi-model system if the selection algorithm is useful in improving the efficiency and effectiveness of the system.

6. Conclusion

This paper investigated an algorithmic linkage between the input images' features and the related CNN model's excitation weights. Based on this feature-weights linkage, we proposed a network selection algorithm, which could choose the appropriate neural network from multiple models for training, verification, and further processing in a complex multi-model system. With this approach, the system could avoid training every model when new objects need to be classified and added to the system. Therefore, by applying this algorithm, the system could enhance its multi-modality learning process with resources saving and less processing time. The proposed algorithm achieved an average selection accuracy of 96.87% in the test of a five-model neural network system.

Although the proposed model selection algorithm could reduce the usage of computation resources and improve the system effectiveness and efficiency, further adaptation and development are required to achieve high selection accuracy for it to be useful in complex multi-model systems. Moreover, it needs, in the future, to be tested adequately to extract the most generalized correct patterns that match the trained models. Nevertheless, this paper provides a new paradigm, which would let machines be capable of developing a multi-model learning technique instead of just building a single large model for future complex neural network systems.

Author Contributions: Conceptualization, EA, XS; methodology, EA, XS, and VDS; software, EA; validation, EA and VDS; formal analysis, EA, XS and VDS; investigation, EA; resources, EA, XS; data curation, EA; writing—original draft preparation, EA, XS; writing—review and editing, EA, XS, and VDS; visualisation, EA, XS; supervision, XS, and VDS; project administration, XS, and VDS; funding acquisition, XS.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kubilius, J.; Schrimpf, M. et al. Brain-Like Object Recognition with High-Performing Shallow Recurrent ANNs. In Proceedings of the *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 10-12 Dec. 2019. pp. 12805–12816. Available online: https://dl.acm.org/doi/pdf/10.5555/3454287 (accessed on 13 Nov 2021)
2. Naseer, M.; Khan, S.; Porikli, F. Indoor Scene Understanding in 2.5/3D for Autonomous Agents - A Survey. *IEEE Access* **2019**, *7*, 1859–1887. doi: 10.1109/ACCESS.2018.2886133
3. Zolanvari, S.M.I.; Ruano, S.; Rana, A.; Cummins, A.; Da Silva, R.E.; Rahbar, M.; Smolic, A. DublinCity: Annotated LiDAR Point Cloud and its Applications. In Proceedings of the 30th British Machine Vision Conference (BMVC 2019), Dublin, United Kingdom, 9-12 Sept. 2019. pp. 1-13. Available online: https://bmvc2019.org/wp-content/uploads/papers/0644-paper.pdf (accessed on 13 Nov. 2021)
4. Li, J.; Yuan, Y.; Mei, K.; Fang, F. Lightweight and Accurate Recursive Fractal Network for Image Super-Resolution. In Proceedings of International Conference on Computer Vision Workshop, Seoul, Korea, 27-28 Oct. 2019. pp. 3814–3823. doi: 10.1109/ICCVW.2019.00474
5. Geetha, G.; Kirthigadevi, T.; Ponsam, G.G.; Karthik, T.; Safa, M. Image Captioning Using Deep Convolutional Neural Networks (CNNs). *Journal of Physics: Conference Series* **2020**, *1712*, 1-13. doi: 10.1088/1742-6596/1712/1/012015
6. Li, K.; Ma, W.; Sajid, U.; Wu, Y.; Wang, G. Object Detection with Convolutional Neural Networks. In *Deep Learning in Computer Vision Principles and Applications*, 1st. ed.; Hassaballah, M., Awad, A.I., Eds.; CRC Press: Boca Raton, FL, USA, 2020; pp. 41-62. doi: 10.1201/9781351003827
7. Contini, E.W.; Goddard, E.; Wardle, S.G. Reaction Times Predict Dynamic Brain Representations Measured with MEG for Only Some Object Categorisation Tasks. *Neuropsychologia* **2021**, *151*, 107687. doi: 10.1016/j.neuropsychologia.2020.107687
8. Zheng M, Kleinberg S. Using Domain Knowledge to Overcome Latent Variables in Causal Inference from Time Series. Proceedings of Machine Learning Research. 2019 Aug; 106:474-489. PMID: 32123870; PMCID: PMC7050445.
9. Hassabis D, Kumaran D, Summerfield C, Botvinick M. Neuroscience-Inspired Artificial Intelligence. Neuron. 2017 Jul 19;95(2):245-258. https://doi.org/10.1016/j.neuron.2017.06.011
10. Pitts, W.; McCulloch, W. The Linear Theory of Neuron Networks: The Static Problem. *The Bulletin of Mathematical Biophysics* **1943**, *4*, 169–175, doi: 10.1007/BF02478112
11. Rawat, W. Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation* **2017**, vol. 29, no. 9, pp. 2352–2449. https://doi.org/10.1162/NECO_a_00990
12. Ullah, A.; Muhammad, K.; Haq, I.U.; Baik, S.W. Action Recognition Using Optimized Deep Autoencoder and CNN for Surveillance Data Streams of Non-Stationary Environments. *Future Generation Computer Systems* **2019**, *96*, 386–397. doi: 10.1016/j.future.2019.01.029
13. Gong, Z.; Zhong, P.; Yu, Y.; Hu, W.; Li, S. A CNN with Multiscale Convolution and Diversified Metric for Hyperspectral Image Classification. *IEEE Trans. Geosci. Remote Sens.*, **2019**, *57(6)*, 3599–3618. doi: 10.1109/TGRS.2018.2886022.

14. Everitt, T.; Hutter, M. Universal artificial intelligence. In *Foundations of trusted autonomy*, 1st ed.; H. A. Abbass, H. A.; Scholz, J.; Reid, D. J., Eds.; Springer Nature: Cham, Switzerland, 2018; Volume 117, pp. 15-46.
15. Joseph, K. J.; Khan, S.; Khan, F. S.; Balasubramanian, V. N. Towards Open World Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20-25 June 2021, pp. 5826-5836, doi: 10.1109/CVPR46437.2021.00577.
16. Kim, H.; Nam, H.; Jung, W.; Lee, J. Performance Analysis of CNN Frameworks for GPUs. In Proceedings of 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Santa Rosa, CA, USA, 24-25 Apr. 2017; pp. 55–64. doi: 10.1109/ISPASS.2017.7975270
17. Kwasigroch, A.; Mikołajczyk, A.; Grochowski, M. Deep Neural Networks Approach to Skin Lesions Classification - A Comparative Analysis. In Proceedings of the 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 28-31 Aug. 2017, pp. 1069-1074. doi: 10.1109/MMAR.2017.8046978
18. Poojary R.; Pai, A. Comparative Study of Model Optimization Techniques in Fine-Tuned CNN Models. In Proceedings of 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 19-21 Nov. 2019, pp. 1-4. doi: 10.1109/ICECTA48151.2019.8959681
19. Hegde, K.; Yu, J.; Agrawal, R.; Yan, M.; Pellauer, M.; Fletcher, C.W. UCNN: Exploiting computational reuse in deep neural networks via weight repetition. In Proceedings of 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture, pp. 674–687. doi: 10.1109/ISCA.2018.00062
20. Quader, N.; Bhuiyan, M. M. I.; Lu, J.; Dai, P.; Li, W. Weight excitation: Built-in attention mechanisms in convolutional neural networks. In Proceeding of the European conference on computer vision, Springer, pp 87–103 2020. doi: 10.1007/978-3-030-58577-8_6
21. Xu, X.; Li, Y.; Wu, G.; Luo, J. Multi-modal deep feature learning for RGB-D object detection. *Pattern Recognition* **2017**, *Volume 72*, pp. 300–313. doi: 10.1016/j.patcog.2017.07.026
22. Long, H.; Chung, Y.; Liu, Z.; Bu, S. Object Detection in Aerial Images Using Feature Fusion Deep Networks. *IEEE Access* **2019,** *Volume 7*, pp 30980-30990. doi: 10.1109/ACCESS.2019.2903422
23. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning Both Weights and Connections for Efficient Neural Networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems, Montreal, Canada, 7 - 12 Dec. 2015, vol. 1, pp. 1135–1143.
24. Jiang, W.; Huang, K.; Geng, J.; Deng, X. Multi-Scale Metric Learning for Few-Shot Learning. *IEEE Transactions on Circuits and Systems for Video Technology* **2021**, *Volume 31*, no. 3, pp. 1091-1102, March 2021, doi: 10.1109/TCSVT.2020.2995754.
25. Lin, T. Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), January 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.
26. Alsaghir, E.; Shi, X.; De Silva, V.; Kondoz, A. Understanding Dilated Mathematical Relationship between Image Features and the Convolutional Neural Network's Learnt Parameters. *Entropy* **2022**, 24, 132. https://doi.org/10.3390/ e24010132
27. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceeding of the 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. International Conference on Learning Representations, ICLR. pp.1-14. https://doi.org/10.48550/arXiv.1409.1556
28. Aggarwal, C.C. Machine Learning with Shallow Neural Networks. In *Neural Networks and Deep Learning* **2018**; Springer: Cham, Switzerland; pp. 455-456. doi: 10.1201/b22400-15
29. Kaggle. Dogs vs. Cats. *Kaggle*, **2013**. Available online: https://www.kaggle.com/c/dogs-vs-cats/data (accessed: 01-Apr-2020)
30. Yasli, B. fruits-360. *Kaggle*, **2020**. Available online: https://www.kaggle.com/barisyasli/fruit360 (accessed: 16-Aug-2020)

31. Dhall, A.; Goecke, R.; Lucey, S.; Gedeon, T. Collecting Large, Richly Annotated Facial-Expression Databases from Movies. *IEEE MultiMedia* **2012**, *Volume 19*, no. 3, pp. 34-41, July-Sept. 2012. doi: 10.1109/MMUL.2012.26.
32. Dhall, A.; Goecke, R.; Joshi, J.; Sikka, K.; Gedeon, T. Emotion Recognition In The Wild Challenge 2014: Baseline, Data and Protocol. *ACM ICMI*, **2014**.
33. Roy, P.; Bhattacharya, S.; Ghosh, S. Natural Images. *Kaggle*, **2018**. Available online: https://www.kaggle.com/prasunroy/natural-images (accessed: 18-Jan-2021).
34. Singh, S. K. Indoor Images. *Kaggle*, **2020**. Available online: https://www.kaggle.com/sshubhamsingh/indoor-images (accessed: 28-Jan-2021).
35. Aggarwal, C.C. *Neural Networks and Deep Learning*; Springer: Cham, Switzerland, 2018; pp. 455-456. doi: 10.1201/b22400-15
36. Shankar, P. M. Pedagogy of Bayes' rule, confusion matrix, transition matrix, and receiver operating characteristics. *Computer Applications in Engineering Education* **2019**, *Volume 27, number 2*, pp. 510 - 518.
37. Tharwat, A. Classification assessment methods. *Applied Computing and Informatics* **2018,** *Volume 17*, no. 1, pp. 168–192. doi: 10.1016/j.aci.2018.08.003
38. Lin, W. and Jay Kuo, C. C. Perceptual visual quality metrics: A survey. *J. Vis. Commun. Image Represent.* 2011, *Volume 22*, no. 4, pp. 297–312, February 2011. http://dx.doi.org/10.1016/j.jvcir.2011.01.005