Image Forgery Detection Based on Deep Transfer Learning

Younis E. Abdalla, M. T. Iqbal, and M. Shehata

***Abstract*—The recent digital revolution has sparked a growing interest in applying convolutional neural networks (CNNs) and deep learning to the field of image forensics. The proposed methods aim to train algorithms for solving a range of predetermined tasks. However, training a model that has been randomly initialized requires extensive time for computation as well as an enormous pool of training data to draw from. Moreover, such a model needs to be developed and redeveloped from the ground up if there are any alterations to the feature-space distribution. In addressing these problems, the present paper proposes a novel approach to training image forgery detection models. The method applies prior knowledge that has been transferred to the new model from previous steganalysis models. Additionally, because CNN models generally perform badly when transferred to other databases, transfer learning accomplished through knowledge transfer allows the model to be easily trained for other databases. The various models are then evaluated using image forgery techniques such as shearing, rotating, and scaling images. The experimental results, which show an image manipulation detection has validation accuracy of over 94.89%, indicate that the proposed transfer learning approach successfully accelerates CNN model convergence but does not improve image quality.**

Keywords—Forgery detection; Deep learning; Transfer learning; Neural network

1. INTRODUCTION

Human beings are generally hard-wired to apply or transfer knowledge that they have learned in one skill to other related skills. In this way, acquired knowledge can help solve problems in less time through the cross-utilization of knowledge which occurs during these transfers [1]. Considering this built-in problem-solving framework, let us look at two critical problems currently restricting progress in the field of image forensics. These problems are: (1) developing a sufficiently large and diverse body of annotated images for use as training models; and (2) including in the models non-image data in order to permit and enable broader application of the model(s).

This research will address both of these challenges. Regarding the creation of a large body of images, this can be an extremely costly undertaking, as the task requires humans rather than machines to do the image classification. Even so, the workers tasked with this job could be severely challenged by the image complexity and vast array of different classes for categorization. Regarding issues around incorporating non-image data in the models, researchers and other workers in the field still experience problems when trying to extract image data from images that also incorporate non-image data, such as text. However, finding a way to include non-image data in transfer learning is crucial if the models are to be applied to areas such as the medical field or insurance industry.

In overcoming these challenges, the present work will employ text metadata to deal with noisy classifications in images. Within these datasets are forged images which will comprise the metadata. So, instead of attempting image classification or categorization, it will be assumed that the images share certain metadata features along with some aspects of feature representations. Two distinct image categories – pristine images and forged images – will be presented, with the metadata being sourced from the Internet. Although this will enable us to substantially expand our available training data set, it will also likely mean that we will be including some forged labels as a trade-off.

We will use cosine similarity to gauge similarities within the metadata, even though this will result in some compromise in feature representation quality and similarity complexity that the system learns. This is another trade-off that is considered negligible. However, there is a relatively significant problem with this strategy – namely, the similarities that exist between the pristine image and forged image metadata. These similarities exist because the Internet-sourced data were not developed as image classifiers but instead were specifically created as fakes of original images.

We note that, though beyond the scope of the present work, numerous applications have shared and related images as well as repeatable features that have similar or even the same colors (e.g., claims in the insurance field that feature damaged vehicle images). These existing models can already incorporate both images and text as data, but they are also prohibitively expensive to build and maintain. This is because large training datasets must be employed to compensate for noisy labels. So, while acknowledging the existence of these models, our work aims to utilize datasets in combination [3], as this approach requires smaller datasets and shows promise of high image quality.

Moreover, our study relies on the concept of transfer learning as a means to reduce both the amount of resources required and the time it takes to train the networks. In our experiments, we initialize the networks by applying basic weights learned in earlier large-scale training, such as CaffeNet and VGG16 [1][13]. We can see by the accuracy in the baseline classification task (i.e., cat versus dog images) that the convolutional neural network (CNN) weights are able to give high-quality feature representation on diverse image datasets. So, our baseline here will be to develop a "cat versus dog" image classifier through fine-tuning CNN weights as a means to significantly shorten the computational time required to train the network.

We will run the network using an extensive sampling of image pairings, with the network learning similarity between the

DOI: http://dx.doi.org/10.24018/ejece.2019.3.5.125

provided images, both forged and pristine. As well, we will fine-tune the CNN weights in order to achieve feature representations with minor differences that are not so different that they require a full re-training. Furthermore, because CNN middle layers and baseline architecture are alike, we aim to demonstrate the practicality and effectiveness of re-training the baseline by applying the original network weights instead of new CNN weights, with the aim of enhancing forgery detection abilities with more accurate image classification.

This paper is organized as follows. Following the introduction, we will provide an overview and revision of the related works. In the subsequent sections, we will dive deeper into the topics mentioned in the overview training, performance testing, and also validation of the used methods.

2. RELATED WORKS

The present paper presents a full review of the background, foundational ideas, and current applications for transfer learning, including both historical and recent examples. It is generally well-known in the industry that transfer learning evolved from machine learning as well as statistical modeling. With this in mind, we look at some research conducted by Han et al. [9] and Doersch et al. [10]. Additionally, we review MatchNet and the study of D. Itera [2], noting that their two-tower architecture shares some similarities with that used in the present work. Specifically, the towers share weights that are first concatenated; the resulting feature representations for patch pairs are then relayed through the metric network (fully connected) and SoftMax loss function. In the metric network, similarities between the two features representations are measured, after which the ground truth similarity loss of the patch pair is formulated. The outcome is equivalent to the present work's formulation for similarities in the two feature representations; however, the ground truth towers rely on the text metadata accompanying the images instead of relying on a computed function overlaying the images.

In [10], Doersch et al. study the behavior of learning features in unsupervised datasets. They divide each image to create a "patch" and then train networks to figure out the right (i.e., original) orientation among the patched pieces. The aim here is to see whether or not a network is able to learn objects occurring within images; if a network can do this, it indicates an ability to learn underlying feature orientation for images. Therefore, by applying these learned representations, features learned from this network could potentially be reused in unsupervised object detection for other datasets.

It is important to note that transfer learning, as mentioned, is a concept which evolved from machine learning and statistical modeling. More recently, transfer learning has been investigated for its application in deep learning. However, earlier approaches that were previously employed to construct and train machine learning models differ significantly from methodologies that adhere to transfer learning strategies. The present work aims to locate similarities in "similar" images' underlying map features by applying the latest deep learning techniques.

3. TRANSFER LEARNING STRATEGIES

The type of transfer learning strategy that is most suitable for a given problem is determined by several factors, including data availability, the task to be performed, and the domain. In general, transfer learning methods are classified according to the kind of conventional ML algorithms that are used. The three main categories explored in this work are: unsupervised transfer learning, transductive transfer learning, and inductive transfer learning, as explained below.

The first category, unsupervised transfer learning, deals with unsupervised tasks located in target domains. Although both the target and the source domains could potentially be similar, their tasks are quite dissimilar. Later in this study, we will work on labeled data which has been made unavailable at both domains. The second category to be explored here is transductive transfer learning, which is employed when there are similarities between target and source tasks but dissimilar corresponding domains. In this scenario, there is no labeled data in the target domain, whereas the source domain contains ample labeled data. Transductive transfer learning can also include subcategories regarding settings (e.g., marginal probabilities or different feature spaces). In inductive transfer learning, which will be the third category studied, both the target and source domains are the same, even though their respective tasks differ. In the inductive transfer learning setting, algorithms employ the source domain's inductive biases as a means to make improvement to the target tasks. This setting can be divided into the sub-categories of self-taught learning and multitask learning, according to whether there is labeled data in the source domain or not. The source task's inductive biases help to carry out the target task. As shown in the figure below, this is accomplished by making adjustments to the target task's inductive biases through the restriction of model space, adjusting the search process using knowledge acquired from the source task, or limiting the hypothesis space.

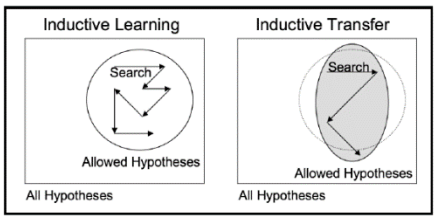

Fig. 1 Inductive transfer techniques of a target [1]

4. TRANSFER LEARNING APPROACHES

We will demonstrate in our work that a few of the approaches are able to be used in the aforementioned strategies, as follows: (1) Instance transfer: This involves reusing (or re-purposing) knowledge obtained from a source domain in a target task. However, in many instances, we cannot directly reuse source domain data, but we might be able to reuse these data in tandem with target data. For inductive transfer cases, we

can apply modifications like AdaBoost (Dai et al.) to assist with improvements to target tasks when training from source domains. (2) Relational-knowledge transfer: This approach deals with non-IID data (e.g., data which are not identically distributed or independent). These type of data have data points that are related to other (sometimes similar) data points. A good example of relational-knowledge transfer in current application is social network data. (3) Parameter transfer: In the parameter transfer approach, it is assumed that models used for related tasks involve a few or more shared parameters and/or hyperparameters. (4) Feature-representation transfer: The purpose of this technique is to mitigate error rates and domain divergence through the identifications of positive (good) feature representations. Such feature representations can then be used for target domains from source domains. Instances where feature-representation transfer is used include supervised/unsupervised methods, if there is sufficient labeled data available.

Similarity detection. In the similarity detection process, similarity is measured by taking any related metadata text applied to an image pair, from which the score of 0 to 1 indicates the similarity of the images, according to the detected features in the objects measured. Although features which are similar will probably produce similar results, repeated features (e.g., neighbourhood or background features) do not always give similar results. Furthermore, this function should be sufficiently robust to compare large and small areas, as patches of various sizes need to be comparable. However, this operation can be very costly, given how many pairs are able to be generated in a dataset and considering that that the operation typically runs several times.

At the outset, the present work employs cosine similarity, which offers users a good similarity measure for comparing sets of varying sizes. However, a trade-off between results accuracy and computational complexity is expected and understood between cosine similarity and other more complicated and time-consuming approaches. Using cosine similarity, two feature "bags" will be developed with feature sets related to the images as well as to the counts for every feature. This model is expected to be biased toward images that are dissimilar, given that dissimilarity has been shown to be more common. As well, we will construct a preprocessing engine which is able to compute image similarity and from that to develop balanced pairs out of the similar/dissimilar images. These will then be labeled accordingly as contradictory categories. Note that this component also includes similarities between different texts and is able to be expanded as needed.

5. SIMILARITY DETECTION

In the similarity detection process, similarity is measured by taking any related metadata text applied to an image pair, from which the score of 0 to 1 indicates the similarity of the images, according to the detected features in the objects measured. Although features which are similar will probably produce similar results, repeated features (e.g., neighbourhood or background features) do not always give similar results. Furthermore, this function should be sufficiently robust to compare large and small areas, as patches of various sizes need to be comparable. However, this operation can be very costly, given how many pairs are able to be generated in a dataset and considering that that the operation typically runs several times.

At the outset, the present work employs cosine similarity, which offers users a good similarity measure for comparing sets of varying sizes. However, a trade-off between results accuracy and computational complexity is expected and understood between cosine similarity and other more complicated and time-consuming approaches. Using cosine similarity, two feature "bags" will be developed with feature sets related to the images as well as to the counts for every feature. This model is expected to be biased toward images that are dissimilar, given that dissimilarity has been shown to be more common. As well, we will construct a preprocessing engine which is able to compute image similarity and from that to develop balanced pairs out of the similar/dissimilar images. These will then be labeled accordingly as contradictory categories. Note that this component also includes similarities between different texts and is able to be expanded as needed.

If we obtain positive results from the simplified similarity function, we would consider carrying out a future study focusing on the kind of trade-offs incurred between using highly complex similarity models and simpler ones, based on effectiveness and runtime. This, however, is outside the scope of the present work, whose main aim is to test similar and non-similar images, not performance grades of similarity.

6. THE PROPOSAL TRANSFER LEARNING APPROACHES

The present work mainly investigates a range of deep learning models. Although the various techniques mentioned above are applicable in different instances of machine learning, is transfer learning also applicable to deep learning? Deep learning models can best be described as being inductive learning approaches. As touched on in a previous section, inductive transfer is a learning mechanism's ability to enhance task performance as a result of having learned a similar skill/task from another (i.e., earlier) one [3]. Hence, a general inductive-learning algorithm objective is to apply mapping based on training examples. So, for example, if the task is classification, a model thus learns to map class labels and input features, using seen and unseen data based on sets of assumptions concerning training data distribution. The assumption sets are referred to as inductive biases and include factors like search process and hypothesis space restrictions. These assumed biases limit the model's learning capacity but also streamline the process. In our proposed approach, we use transfer knowledge in model image classification related to the categories of "cat" and "dog", using a pre-trained model for detecting pristine images in comparison to forged images.

Datasets Environment

This experiment was conducted using multiple datasets [3]. The training used mainly dogs vs cats dataset which 25,000 images of dogs and cats in total each category has 12500 images. We can verify with the preceding output that we have 12,500 images for each category. Let's now build our smaller dataset, so that we have 3,000 images for training, 1,000 images for validation, and 1,000 images for our test dataset and that applied to the both categories [4].

A collection dataset out of online and public existing datasets for training and testing. In total, we collected 1792 pair

images with good quality to present different samples for copy-move forgery. The first one was constructed by Christlein et al [5], consisting of 48 base images and 87 copied with a total of copy-move forged images of 1392. The second database MICC-F600 was introduced by Amerini et al [6] [7] with 400 images. Caltech-101, image manipulation dataset. IM dataset (240 images) [5]. The Oxford buildings dataset (198 images and 5062 resized images) [8]. Coverage dataset (200) [9] and collection of online and self-producer images. Note that, even the total images look bigger than what we used in training and testing, that is because we avoid using some images either because they are in bad shape or low resolution. Therefore, here we used selected ponch of images with total of 1348 image which devided for 1248 for training and 100 for testing task.

7. IMPLEMENTATION

Using Python 3.6, our experiments ran basic classification task on a network of similar images pairs in MatchNet. The model training data were kept in a secure H5 file, and the original convolutional neural network was built accord to [4]. Hence, the transfer learning network is more or less identical to the earlier network that employed pre-knowledge from the original CNN model. The towers have two main requirements, as follows. 1) Weight share: The towers have to be able to share weights, as this is how the network is able to learn image pairs simultaneously (i.e., running image pairs through the same weight set for every layer). 2) Towers as CNNs: Specifically, the towers must be CNNs of themselves, which then permits the learned weights from one network model to be used on the other. Therefore, the towers must have the same parameter and architecture names to enable weights to be transferable between them.

This is accomplished as follows. In the weight-training stage, weights are initialized. At the completion of the training, the model is saved in an H5 file, which can then be loaded for additional training/testing utilizing the same weight parameters. The CNN networks comprise standard convolutions, normalization layers, ReLu and pool (i.e., frozen convolutional layers). However, they are unique in that images that are loaded are split, after which they are recombined prior to reaching fully connected layers. Individual images are fed into input layers, with the outputs being vectors with representations comprising 4,096 elements. Next, the vector is fed to fully connected layer sets which have been re-trained to convert vectors into similarity/non-similarity scores. In the final step, the cosine similarity computes a Softmax loss of expected similarity vs. predicted similarity for image content. The total operation is function as deep learning classifier to classify the output in the designation target.

8. EXPERIMENT RESULTS

There are several tasks that can be evaluated using these features to compare to the state-of-the-art systems. While there are many such tasks in the unsupervised learning space, to bound the difficulty of evaluating the results of this work, the first set of experiments will simply attempt to test if the weights learned in the CNN are a better initialization for fine-tuning an image classification task than the provided weights which were learned on the training task. The rest of this section will outline in detail how these experiments were built and how weights were transferred.
Frist, we present the result of training the original model to show how the results look like and then we'll use the new dataset to see how the knowledge transferred and show the result for the new task.

The baseline training model and the transfer learning model are layered in same architectures. The last layers (dense layers) were trained slightly different to serve the different output target. This architecture of layered network allowed us to employ the trained/or pre-trained network in the original model to extract the new features in the other model with the consideration of re-train the classifier to serve the new labeled case. Figure 2 show the models summery in both tasks while the table 1 shows the parameters' summary of the same model.

Table. 1 Model summary information

Model parameters summary	Total parameters	Trainable parameters	Non-trainable parameters
	12,942,786	12,941,314	1,472

In this work, the initial experiments aim to determine whether the weights that were learned in the CNNs improve initialization of image classification task fine-tuning compared to the weights learned during training task of the original dataset. The remainder of the section provides a detailed discussion on the construction of the experiments as well as the manner in which the weights are transferred during the tasks. Specifically, we will review and present the outcomes from the original model training, after which we will apply the new dataset in order to determine the means of knowledge transfer. The results of the new approach will also be provided.

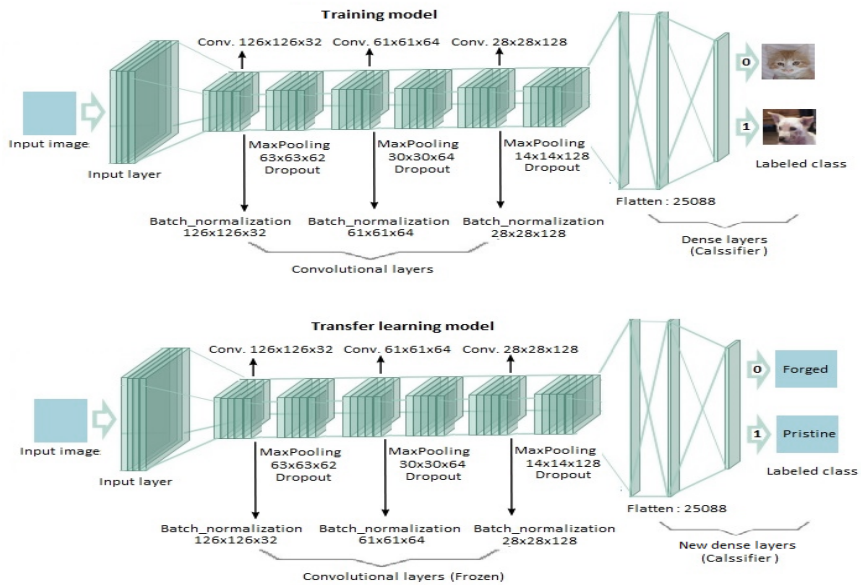

Fig. 2 Shows the summary of the processing of the transfer learning model form the original trained model

Training data generator

In normal cases the all dataset images can't be uploaded to the process memory in one shout. Also, this may drop down the performance of the used GPU's / or CPU in the training task. Therefore, we will load the dataset in group of 15 images at once in each time to all dataset using data generator. The other advantage of using data generator is make many changes in each image which known as data augmentation to learn all possible image transformation to the used neural network after fixing the image scale according to the input layer which in our case is $126 \times 126 \times 32$. Also this will present different image forgery techniques such as shearing, rotating, and scaling images. Figure 3 shows two examples of data generator. The output of this task can be summarized as following: Found 20000 images belonging to 2 classes. Training Generator: Found 5000 images belonging to 2 classes. Found 0 images belonging to 0 classes.

Fig. 3 Shows examples of the data generator work using the original dataset

Note: Categories: 0: 'cat', 1: 'dog'. The original dataset has same number of the two categories that used in the training task as shown in the next illustration, figure 3.

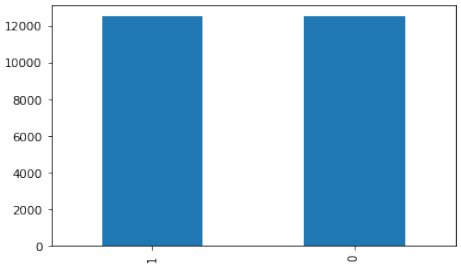

Fig. 3 shows the training result map to dog is 1 and cat is 0

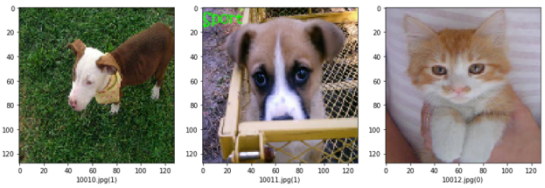

Fig. 4 Random samples from the training result in image's presentation

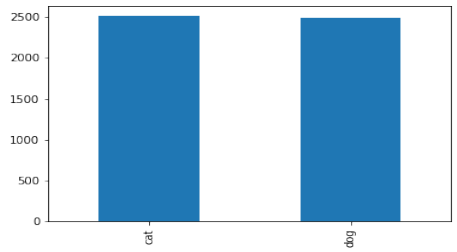

Fig. 5 Shows the validation result map to model using the original dataset: dog is 1 and cat is 0

Create Testing Generator

We will convert the predict category back into our generator classes by using train_generator. class_indices. It is the classes that image generator map while converting data into computer vision

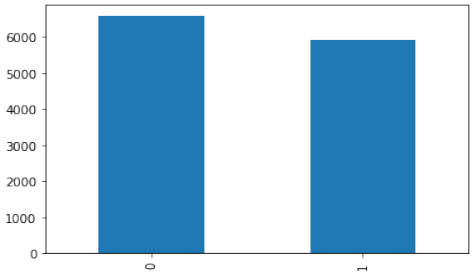

Fig. 6 Shows the testing result map to dog is 1 and cat is 0

Fig. 7 Shows the predicted result with images representation

Training task initial output: Found 998images belonging to class 1, found 250 images belonging to class 0.
After preparing the new dataset and go through the above steps we see the results:

Fig. 8 Random sample images from the new dataset

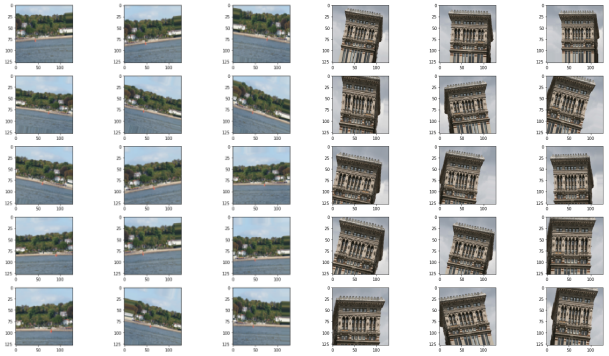

Fig. 9 Shows examples of the data generator work using new dataset

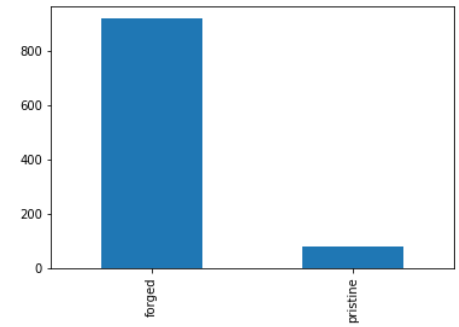

Fig. 10 Shows a map result of training set for new dataset

Table 2 Shows the numerical raining result which was presented in the above figure

	category	filename		category	filename
0	0	forged (1).png	1243	1	pristine (92).png
1	0	forged (10).png	1244	1	pristine (93).png
2	0	forged (100).png	1245	1	pristine (94).png
3	0	forged (1000).png	1246	1	pristine (95).png
4	0	forged (1001).png	1247	1	pristine (96).png

For the training task, we used the two categories with total image of 1248 images as we mentioned above. Here we show random sample pairs images out of the used dataset:

Fig. 11 Shows random samples of the two new categories with their labels

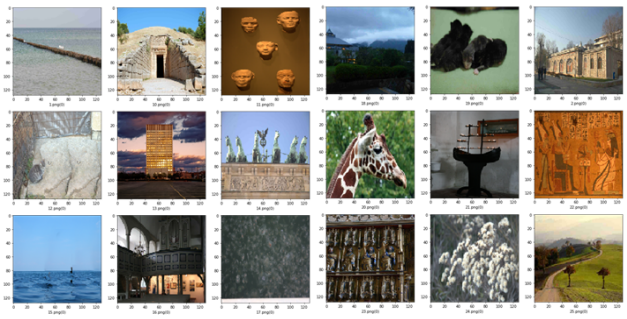

Fig. 12 Shows the predicted result with images using new dataset

Now for the new dataset the result should look like:
('dog': 1, 'cat': 0) => ('pristine': 1, 'forged': 0)
i.e. each image labeled with dog that means this image is a pristine image, and each image tagged in cat is a forged image as you can see in figurre13.

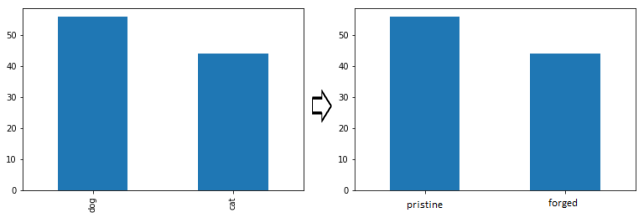

Fig. 13 Shows the image labels presentation for new dataset based on the original dataset labels.

Based on the original model weights the second dataset will be judged. Next figure shows the deep learning classifier output after re-trained to deserve the new task. However, the output still shows some false detection which can be overtaken by using closer model for the learning transfer assignment.

Fig. 14 Shows some predicted results in images presentation for validation task

According to sample result presented in the figure 14, the first image from left hand is forged image and here shows as (dog==pristine). The other two images are (cat==forged) and that is correct.
Now, in order to increase the accuracy of this model, we train the original datatset for longer time using 25 Epochs with same conditions to avoid over fitting.

Validation: The next figure 15 shows the validation data for both datasts before we flatten the data and feed our deep learning classifier.

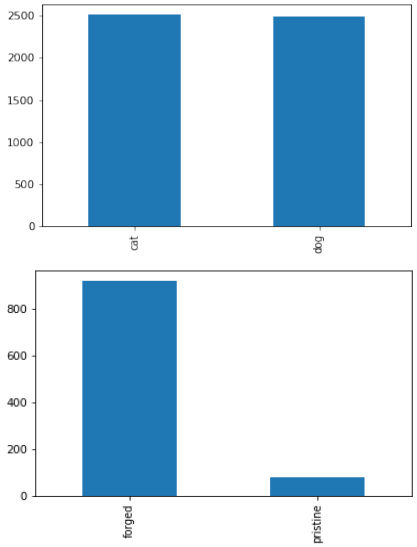

Fig. 15 Shows the validation data of the presented models in both datasets.

As shown in the results, the validation accuracy of the baselines was higher than the accuracies using the weights learned in the original network. The next visual illustration shows clearly the gap between the training accuracy and validation accuracy for the both datasets.

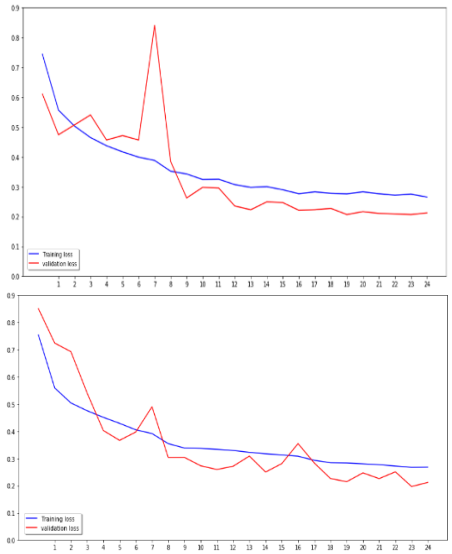

Fig. 16 Shows the training loss vs. validation loss in different training trial

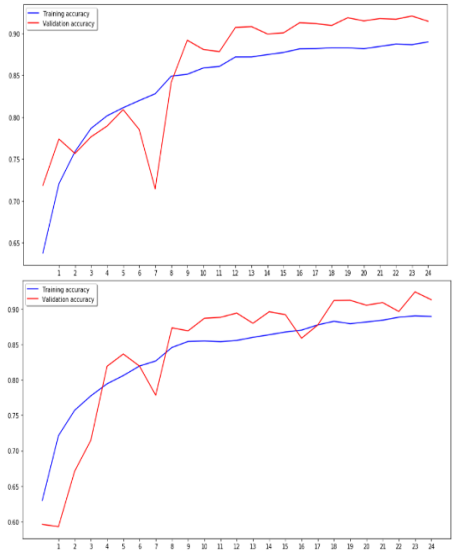

Fig. 17 Shows the training accuracy vs. validation accuracy in different training trial

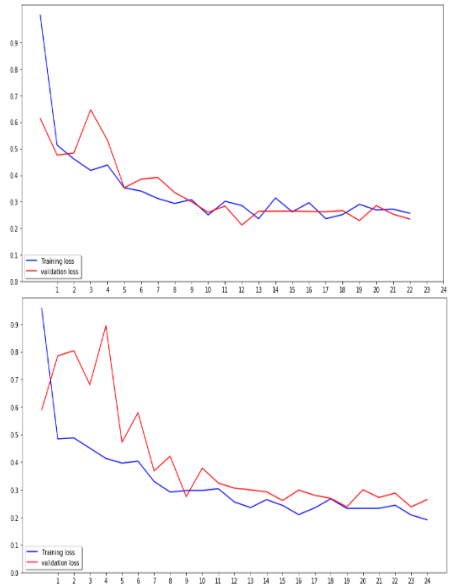

Fig. 18 Shows the training loss vs. validation loss in different training trial (new dataset)

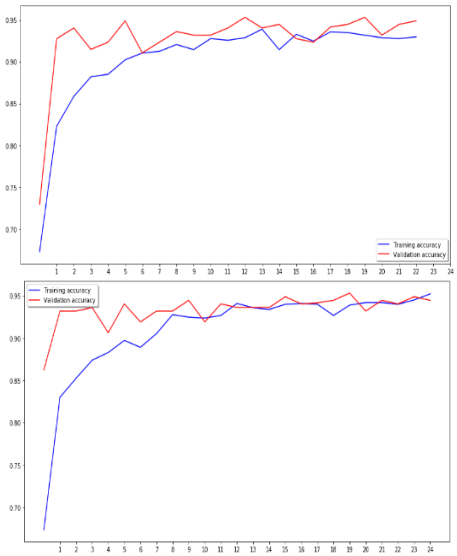

Fig. 19 Shows the training accuracy vs. validation accuracy in different training trial (new dataset)

From the figures 16, 17, 18 and 19 we can notes that the model is getting over fitting with new dataset, however, the model still able to maintain excellent validation accuracy. However, second training show more improvement in both loss and accuracy training vs. validations. Also, from the table 3 we can see that we achieved validation accuracy of 94.89% which 4% improvement in accuracy from the previous model.

Table. 3 Shows the loss and the validation accuracy of the both models

The model	Loss	Acc.	Val loss	Val Acc.
Trained model	0.2683	88.93%	0.2122	91.29%
Transfer model	0.2583	92.94%	0.2347	94.89%

Evaluate this work with the-state-of-the-art is presented in the table 4.

Table. 4 The evaluation based on the validation accuracy between two closer targets

The Model	Training data	Validation accuracy
[2]	MSCOCO	77%
	ImageNet subset	93%
Presented model	Dogs & Cats	91.29%
	Pristine & Forged	94.89%

9. Conclusion and Future Work

The present work demonstrates that, by employing different CNN architectures, deep learning can be successfully applied in tasks such as image classification, image identification and object recognition. Cost-effective image classification is achieved on manipulated and/or larger datasets, and improved image feature mapping are obtained from similar images in text metadata using CNNs. However, although using feature map representations is shown to be cheaper and faster, it does not improve the quality of the image classifications, indicating that this approach is not optimal for evaluating quality, given the weak correlation between feature labels and similar (and/or non-) images. Nevertheless, the results of the present work could lead to future investigations that include looking at other forms of forgery detection by applying the newly transfer learned weights. Overall, the present work indicates that metadata sampling and classification requires highly disciplined scaling model which can be scored by employing pre-trained model with and that can be a future extension steps to this work.

Acknowledgement

This work is funded by the ministry of higher education of Libyan Government, which managed by CBIE in Canada. The authors acknowledged Mr. T. Iqbal, who make most of the help and reviews to make this work existed.

Author's Contributions

The authors confirm that the manuscript has been read and approved by all authors and that there are no other persons who satisfied the criteria for authorship but are not listed.

Conflict of Interest

The authors declare that there is no actual or potential conflict of interest regarding the publication of this article.

References

[1] R. T. C. Dipanjan Sarkar, Hands On Transfer Learning With Python, Packt, 2018.
[2] D. Iter, "Image Classification using Transfer Learning from Siamese Networks based on Text Metadata Similarity," Stanford University, pp. 1 - 13, 2016.
[3] R. V. G. B. Soares, " Inductive Transfer," In: Sammut C., Webb G.I.(eds) Encyclopedia of Machine Learning. Springer, Boston, MA, 2011.
[4] D. Sarkar, "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning," *Midum,* 14 Nov. 2018.
[5] W. H. Z. Jing, "Exposing digital forgeries by detecting traces of image splicing," *in 8th International Conference on Signal Processing. IEEE,* vol. 2, 2006.
[6] A. V. A. Mahendran, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision,* vol. 12, no. 3, pp. 233-255, 2016.
[7] K. Z. and He, "Identity mappings in deep residual networks," *arXiv preprint arXiv,* no. 1611.05431, 2016.
[8] R. A. James P., "http://robots.ox.ac.uk/~vgg/data/oxbuildings/," [Online]. [Accessed 8 Aug. 2019].
[9] Y. Z. B. Wen, "COVERAGE - A Novel Database for Copy-Move Forgery Detection," *in Proc. IEEE Int. Conf. Image Processing (ICIP),* pp. 1-19, 2016.
[10] T. A. B. X. Han, " MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching.," *Proceedings of Computer Vision and Pattern Recognition,* 2015.
[11] M. T. Lin, "Microsoft COCO: Common Objects in Context," in *ECCV*, 2014.
[12] D. A. B. Thomee, "The New Data and New Challenges in Multimedia Research," *arXiv* , 2015.
[13] E. S. Y. Jia, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv* , 2014