UDC 004.932.2

F. Smilianets

APPLICATION OF EMBEDDINGS FOR MULTI-CLASS CLASSIFICATION WITH OPTIONAL EXTENDABILITY

Abstract: This study investigates the feasibility of an expandable image classification method, utilizing a convolutional neural network to generate embeddings for use with simpler machine learning algorithms. The possibility of utilizing this approach to add new classes by additional training without modifying the topology of the vectorization network was shown on two datasets: MNIST and Fashion-MNIST. The findings indicate that this approach can reduce retraining time and complexity, particularly for more complex image classification tasks, and also offers additional capabilities such as similarity search in vector databases. However, for simpler tasks, conventional classification networks remain more time-efficient.

Keywords: multiclass classification, convolutional neural networks, embeddings, embedding-based classification, image classification.

Introduction

Image classification is a well-established domain of machine learning. Countless methods and approaches are researched, implemented and used to solve real-world problems. Unfortunately, a typical image classification solution is fairly rigid, in the sense that to add a new class it would require topological change and significant retraining from scratch. But, some use-cases – like disease classification – might benefit from a solution which would be designed in a way to make the extension process simpler and quicker.

This paper is describing the feasibility of using an already existing image classifier to create an expandable image classification process, relying on a convolutional neural network tasked with producing embeddings of given images, for these embeddings to be used in classification by simpler quickly retrainable machine learning algorithms like Random Forest Classifier [1] or KNeighboursClassifier [2]. Additionally, produced embeddings could be employed for similarity search in a vector database.

The intended benefit of such an approach is the ability to add additional classes without having to topologically modify the part that takes the most time to train – the deep neural network. Instead, support for additional classes is enabled by adding new data into the pool and fine-tuning an existing model, previously trained on earlier classes, while the simpler algorithms classifying on embeddings could be retrained from scratch in insignificant time.

 ISSN 1560-8956

Research goal

The goal of this research is to prove that adding additional classes without change to the topology of the vectorizer network is possible while maintaining acceptable classification accuracy on the produced embeddings, using readily available general image classification datasets.

Datasets

To test the proposed method, two datasets were chosen, described in Table 1.

Table 1.

Datasets used

Parameters	MNIST [3]	Fashion-MNIST [4]
Image size	28x28	28x28
Image color	Grayscale	Grayscale
Image content	Hand-written digits	Clothing articles
Full dataset size (train/test)	60000/10000	60000/10000
3-class subset size (train/test)	18623/3147	18000/3000
4-class subset size (train/test)	24754/4157	24000/4000
Classes used	0, 1, 2 and 3	t-shirt/top, trouser, pullover and dress

Since this paper is aimed at testing classifier expandability, only subsets of the datasets were used – a 3-class subset to train the initial classification method, which are then expanded with a 4^{th} class.

First dataset is a standard image classification dataset, MNIST, which is a reorganized subset of a larger set by NIST. An example of it is presented on Fig. 1.

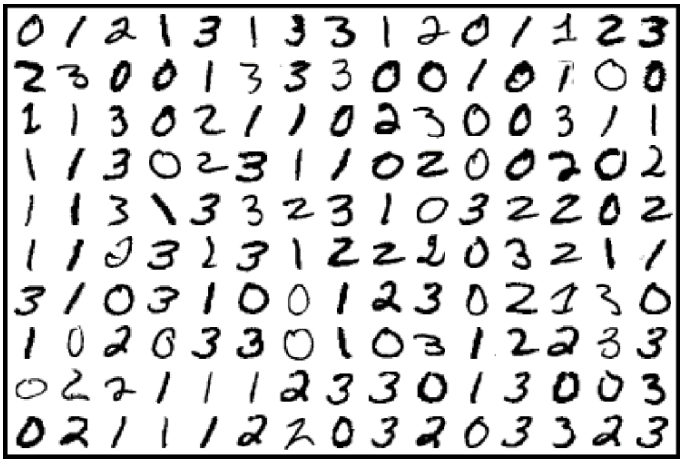

Figure 1. Examples of images from MNIST dataset (color inverted)

Second dataset, Fashion-MNIST [4] (FMNIST), is a drop-in replacement for MNIST, developed by Zalando to provide a better benchmark for classification algorithms.

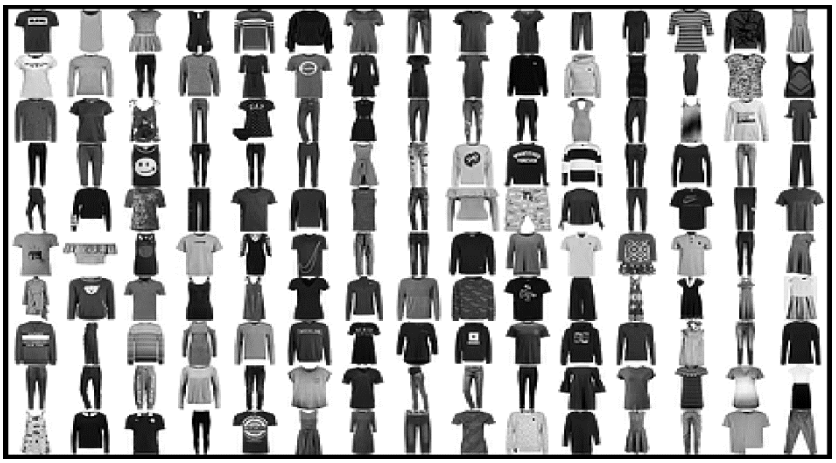

Figure 2. Examples of images from Fashion-MNIST dataset (color inverted).

As evident from the examples, FMNIST presents a substantially harder challenge for image classification algorithms, and is considered more representative of modern computer vision tasks.

Training and testing the networks

The process of testing the stated hypothesis consists of following steps:

1. training a classical convolutional neural network for classification;
2. using the weights of the convolutional part of the neural network (step 1) to train an embedding generator followed by training simpler algorithms for classification based on these embeddings;
3. adding the fourth class to the dataset and retraining the embedding generator, followed by training simpler algorithms for classification based on these embeddings;

As a first step, a standard convolutional network for image classification was created, modelling after an example network, used by Rahimzadeh, Attar, and Sakhaei in [5] to classify Computed Tomography lung images to differentiate between a COVID-19 pneumonia and healthy lungs. The resulting network is shown on Fig. 3.

This network consists of three principal sections. The input data is being processed by a standard ResNet50V2. Data from both end layer and inner layers of the ResNet is extracted and reprocessed by a Feature Pyramid Network [6] (FPN) with the purpose of aiding with detection of features of various size and abstractness, from details of texture to general shapes. Features, extracted by the FPN are then used by Dense layers to make the final classification decision.

For this deep network to be functional, a Keras Resizing layer was inserted before the ResNet inputs, upscaling the input images from 28x28 to 128x128, so that the network would be able to convolve on all the layers without running out of dimensions.

This network was trained on 3 classes of MNIST during 4 epoches using Nadam with a learning rate of 1×10^{-4} with Categorical Crossentropy as the loss function. Additionally, a

 ISSN 1560-8956

clone of this network with same exact hyperparameters was trained on Fashion-MNIST was trained for 8 epoches.

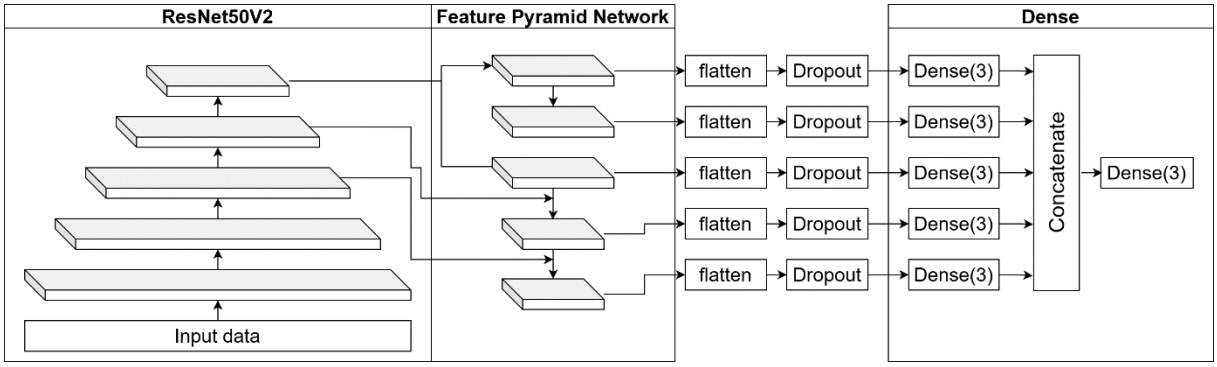

Figure 3. Diagram of three-class classification network

As a second step, he achieved networks were then used to construct an embedding generator by replacing the end Dense layers doing the classification with a single Dense layer which would output a vector of 64 elements. The result of such a modification is displayed in Fig. 4.

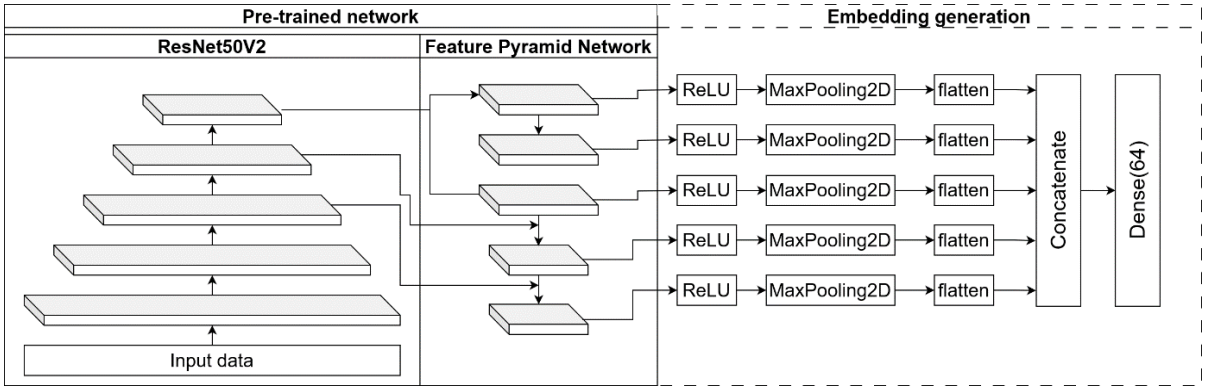

Figure 4. Diagram of resulting embedding generation network

As evident from the illustration, the outputs of the FPN were stratified with ReLU to accentuate found features, the dimensionality was then reduced using MaxPooling2D, and the resulting data was then flattened and concatenated into an input for the Dense layer, forming the embedding.

To achieve meaningful embeddings, the network must be incentivized to keep embeddings of images of different classes apart, which would need to be achieved without actually showing the classes to the network. For that, the training process and data batches need to be constructed in a way for the loss function to compare outputs of the network in training generated with samples of different classes. The loss function itself would need to stimulate distinction between embeddings of different classes. Therefore, a procedure from [7] was adapted with some modifications.

The method, described in [7] assembles a data batch which consists of a set of triplets containing anchor, positive (an image of the same class as anchor) and negative (image of a different class from the anchor) images, chosen randomly, with the intention for the anchor to

get closer to positive and farther from negative during training. Each class presented in the dataset is represented as an anchor only once per batch. Such an approach yielded poor results due to the batches being too small, so it was decided to introduce a batch_class_repeats hyperparameter, which would regulate how many times in a given batch each class is used as an anchor in a triplet, allowing indirect adjustment of batch size.

The training approach in [7] computes the embeddings for each of the anchor, positive and negative images, and with those, computes cosine similarities in anchor-positive and anchor-negative pairs (1):

$$cosinesimilarity = \frac{A \cdot B}{\|A\|\|B\|}, \quad (1)$$

where A and B are the embeddings of compared images. The computed similarities are then scaled down with a coefficient of $k = 0.2$ (2):

$$scaledsimilarity = cosinesimilarity * k. \quad (2)$$

Afterwards, the similarities are used to compute a Sparse Categorical Crossentropy loss function against class labels to bring anchor-positive pairs closer together and all the other further apart. This approach did not yield good results either. Thus, the training procedure was rewritten to use the cosine similarities in a Triplet Loss [8]. Final form of the loss function is the following (3):

$$loss = max(P - N + m, 0), \quad (3)$$

where P and N represent positive and negative cosine similarities respectively, and m is the margin, by which the vectors should be separated. Triplet loss was chosen specifically due to the explicit nature of forcing separation of different classes.

The embedding networks for both MNIST and FMNIST were trained using the described approach with the following parameters:

- batch_class_repeats=15;
- $k = 0.2$ similarity scaling factor;
- $lr = 1 \times 10 - 3$ learning rate;

using a Nadam classifier until reaching a learning rate lower than 5000. During training, the convolution layers, borrowed from straight classification networks were used as static (frozen) to utilize knowledge, attained during training as part of a classification network.

With trained embedding generation networks, both the training and testing parts of the datasets were encoded into embeddings, and a series of simpler machine learning classification algorithms were trained and evaluated.

As a third step, to test the expandability hypothesis, fourth class was added from the respective dataset into the training data for both networks. The networks then received additional training on the expanded dataset, after which the embeddings for the entire dataset were re-generated, and a set of classic machine learning algorithms was once again trained on the re-generated embeddings.

For each of four achieved embedding-generated networks, the images from their training data were vectorized using the network, and on each such set of vectors, a set of classic machine learning algorithms was trained.

 ISSN 1560-8956

Results and discussion

All of the trained networks were evaluated on the portions of test sets of the respective datasets, containing only the classes that were used in training for the respective network. For embedding-based solutions, the test set pictures were vectorized into embeddings and classic machine learning algorithms were evaluated using these vectorisations.

The straight 3-class classifier convolutional networks achieved an accuracy of 0.998 for both MNIST and Fashion-MNIST. Accuracies for machine learning classification algorithms, trained on embeddings produced with the starting 3-class embedding model and the extended 4-class embedding model are displayed in Table 2.

Table 2.

Accuracies for embedding-based classification.

Method	3-class accuracy, MNIST/F-MNIST	4-class accuracy, MNIST/F-MNIST
SVC	0.997/0.977	0.987/0.9435
KNeighborsClassifier	0.999/0.978	0.988/0.932
3-class embedding, Random Forest	0.997/0.979	0.988/0.944

The experiments were done without any hyperparameter tuning or other corrections that could make the classification accuracy with the expanded embeddings. The retraining for the 4-class based generators was also done in the same manner as was the initial training.

Thus, while these results can be improved upon by changing the training algorithm for the embedding network and tuning various hyperparameters, the results shown in Table 1 do indeed prove that such an approach is able to yield classification systems which are expandable without loss of accuracy in scale which would render the system unusable.

Let us examine the time costs for initial training of a common classification network (4):

$$t_{com.f.} = t_{d.p.} + t_{imp.} + t_{tr.}, \tag{4}$$

where:

- $t_{com.f.}$ is the full time spent to achieve a common classification network;
- $t_{d.p.}$is the time to form and preprocess the dataset;
- $t_{imp.}$ is the implementation time for the neural network;
- $t_{tr.}$ is the training time for the neural network.

In comparison, using the embedding-based approach yields additional time components (5):

$$t_{emb.f.} = t_{d.p.} + t_{imp.} + t_{tr.} + t_{imp.c.} + t_{tr.c.}, \tag{5}$$

where:

- $t_{emb.f.}$ is the full time spent to achieve an embedding-based classificator;

- $t_{imp.c.}$is the implementation time for the classifiers taking embeddings as their input;
- $t_{tr.c.}$ is the training time for the classifiers taking embeddings as their input (is significantly smaller than the neural network training time, and can be neglected).

Let us examine the time costs to add a new class to a common classification network (6):

$$t_{com.ex.f.} = t_{a.d.} + t_{mod.} + t_{tr.}, \tag{6}$$

where:

- $t_{com.ex.f.}$ is the full time spent to achieve a common classification network expanded with a new class;
- $t_{a.d.}$is the time to add the new class to the dataset;
- $t_{tr.}$ is the time to train the neural network.

Instead, when using the embedding-based approach, no code modification is necessary, which leads to next time costs (7):

$$t_{emb.ex.f.} = t_{a.d.} + t_{add.tr.} + t_{tr.c.}, \tag{7}$$

where:

- $t_{emb.ex.f.}$ is the full time spent to achieve an embedding-based classification method expanded with a new class;
- $t_{add.tr.}$ is the time for additional training of the neural network.
- $t_{tr.c.}$ is the training time for the classifiers taking embeddings as their input.

Provided that for a practical task $t_{add.tr.} \leq t_{tr.}$ and $t_{tr.c.} \ll t_{mod.}$, the total time to add a new class to an embedding-based classifier would be significantly smaller than for a common classifier.

Therefore, since achieving an extended classifier using embeddings requires human involvement only on the data addition step, other steps can be automated, which, in theory, allows to update faster to support new classes.

Conclusions

The embedding-based approach, presented on the examples given, shows that it is applicable to classification problems, and allows for extension with new classes without any changes to network topology. For simpler tasks, such as MNIST, the time costs for implementing and training the discussed approach do not justify the stated advantages, as training conventional classification neural networks is comparatively faster for such tasks. However, for more complex tasks involving larger and more complex images, this approach may be desirable because it allows for retraining the neural network without changing its topology. This can be crucial in scenarios where the rapid addition of new classes is important. The discussed approach also provides additional capabilities, such as using the obtained embeddings in vector databases for similarity search.

REFERENCES

1. *Breiman L*. Random forests / L. Breiman // Machine Learning. – 2001. – Vol. 45, No. 1. – P. 5–32.

2. *Cunningham P*. K-nearest neighbour classifiers - a tutorial / P. Cunningham, S. J. Delany // ACM Computing Surveys. – 2022. – Vol. 54, No. 6. – P. 1–25.

3. *LeCun Y*. The mnist database of handwritten digits / Y. LeCun, C. Cortes, C. J. C. Burges. – 1998.

4. *Xiao H*. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms / H. Xiao, K. Rasul, R. Vollgraf // 2017.

5. *Rahimzadeh M*. A fully automated deep learning-based network for detecting covid-19 from a new and large lung ct scan dataset / M. Rahimzadeh, A. Attar, S. M. Sakhaei // Biomedical Signal Processing and Control. – 2021. – Vol. 68. – P. 102588.

6. *Lin T.-Y*. Feature pyramid networks for object detection / T.-Y. Lin, P. Dollár, R. Girshick, [et al.]. – 2017.

7. *Kelcey M*. Metric learning for image similarity search / M. Kelcey. – 2020.

8. *Hoffer E*. Deep metric learning using triplet network / E. Hoffer, N. Ailon. – 2018.