# Convolutional Neural Networks for Subfigure Classification

David Lyndon[1], Ashnil Kumar[1,3], Jinman Kim[1,3], Philip H. W. Leong[2,3], and Dagan Feng[1,3]

[1] School of Information Technologies, University of Sydney, Australia
[2] School of Electrical and Information Engineering, University of Sydney, Australia
[3] Institute of Biomedical Engineering and Technology, University of Sydney, Australia
dlyn9602@uni.sydney.edu.au
{ashnil.kumar,jinman.kim,philip.leong,dagan.feng}@sydney.edu.au

**Abstract.** A major challenge for Medical Image Retrieval (MIR) is the discovery of relationships between low-level image features (intensity, gradient, texture, etc.) and high-level semantics such as modality, anatomy or pathology. Convolutional Neural Networks (CNNs) have been shown to have an inherent ability to automatically extract hierarchical representations from raw data. Their successful application in a variety of generalised imaging tasks suggests great potential for MIR. However, a major hurdle to their deployment in the medical domain is the relative lack of robust training corpora when compared to general imaging benchmarks such as ImageNET and CIFAR. In this paper, we present the adaptation of CNNs to the subfigure classification subtask of the medical classification task at ImageCLEF 2015.

**Keywords:** Deep Learning, Convolutional Neural Networks, Medical Image Retrieval

## 1 Introduction

This paper documents the Biomedical Engineering and Technology (BMET) team from the University of Sydney's submissions for the ImageCLEF 2015 [1] Medical Classification task [2]. Specifically, BMET's work was directed at the Subfigure Modality Classification subtask.

The objective of our experiments was to evaluate the effectiveness of Convolutional Neural Networks (CNNs) for this subtask. In particular, we propose a deep learning framework that could learns high-level representations of different image modalities and use these to classify the modality of each subfigure.

## 2 Background

Convolutional Neural Networks, a type of deep learning algorithm, have been used to produce state-of-the-art results for a variety of machine learning tasks

such as image recognition, acoustic recognition and natural language processing since 2012 [3–5]. CNNs share the common features of all deep learning algorithms: stacked layers of neuronal subunits that learn hierarchical representations (allowing the data to be understood at various levels of abstraction, in isolation or combination [3]), the ability to perform unsupervised pre-training on unlabeled data and efficient parallelization on multiple core GPUs which can result in improvements of up to 5000% over CPU-only implementations [4].

A more subtle implication of deep learning is that it can automatically extract features from raw data [3–5]. Typically, a key factor in the success of typical machine learning algorithms is extracting salient features from the raw data. Taking image recognition as an example, a feature set such as edges or SIFT [6] would be extracted from the raw data and it is these new features per se or in combination with the original raw data that would be fed into the machine learning algorithm. While some aspects of the process can be automated or implemented with well known algorithms, a major drawback is that it generally requires expert domain knowledge to define which features should be used and evaluate their success.

Deep learning algorithms, however, are able to directly utilise raw data instead of hand-crafted features. By feeding the data sequentially through many successive layers of subunits, the higher levels of the system are able to understand the data in terms of successively abstract representations [3].

Medical Image Retrieval (MIR) tasks, such as the tests devised for ImageCLEF, require learning precisely these kinds of highly abstract representations, i.e. image modality or the anatomical semantics of the image. However, to the best of our knowledge it is not currently a well established method in this domain. This is due to not only the inherent challenges of medical images[7], but also because state-of-the-art deep learning results are typically obtained using huge sets of labelled training data[4] on tasks that are arguably less subtle. As a justification for these claims, consider that the ImageNET general object recognition task corpora consists of millions of robustly labelled images and was created with the assistance of crowdsourcing via Amazon Mechanical Turk [9]. On the other hand, medical imaging datasets require careful labelling by domain experts, often specialists in a particular area [7, 10, 11] and as a result are generally much smaller.

Large training sets are a current necessity of very deep systems because they contain many millions of internal parameters that must be estimated from the data. Too little data can result in the the higher-level neurons' activation being the result of salient features of the training set and not reflecting the high-level representations. If this 'overfitting' occurs then the system's ability to generalise on new data is severely impaired [12].

In addition to the issues regarding the volume of data required, it must be mentioned that while deep learning can automatically perform excellent feature extraction, this comes at the significant cost of the larger number of hyperpa-

[4] Krizhevsky et. al. [8] used approximately 1.2 million labelled examples for their breakthrough result in ImageNET in 2012.

rameters that must be evaluated in order to find an optimal system [13]. For example, compared to a commonly used machine learning algorithm such as the Support Vector Machine (SVM) that has a basic hyperparameter search space with dimensions of choice of kernel, regularization constant and kernel hyperparameter, even the simplest implementation of a CNN requires fundamental choices about the number and type of layers, filter size and number of filters per layer, and the learning rate. More advanced implementations include factors such as unit activation function and the use of dropout. While there are guidelines for these choices in the literature [13], the difficulty of even a small parameter search is compounded by the increased computational requirements of training the system.

## 3 Methods

### 3.1 Image Preprocessing

A requirement of our classifiers was uniformly sized input vectors, however, the supplied training data varied greatly in size. This was achieved by square-cropping the image to 500px, any dimension of the image smaller than 500px was filled with black pixels.

Even prior to training the CNN, we were aware that the computational requirements were quite demanding and this would be exacerbated by using large images. We resized the images to 160x160px to reduce the computational overhead that would have been required by using higher resolution images. Good results have been reported in the literature for complex tasks with 48x48px images [14] and Krizhevsky et. al. [8] achieved state-of-the art general object recognition with 256x256px images (technically, the system had an input of 224x224px, but these were subimages of the original 256x256px images).

After resizing the images were 160x160x3px, the third dimension describing the three colour channels. For the purposes of simplicity and to further reduce the computational requirements we reduced the 3 channel colour representation to a single channel (red).

We randomly divided the training data into a 70/30 split for training and validation.

### 3.2 Softmax Classification

We evaluated the effectiveness of CNN-derived features by comparing it to the results achieved by a Softmax classifier on the raw data. This experiment is important because the CNN's final layer is the input to a Softmax classifier. This experiment can therefore be used to quantify the effectiveness of the unsupervised feature extraction performed by the CNN.

### 3.3 Convolutional Neural Network

The architecture for the CNN used for our experimentation was based on a simplified version of Yann LeCun et. al.'s [15] LeNet-5[5]. This CNN is capable of correctly classifying the MNIST handwritten digit database with 1.7% test error. We modified the input to account for larger images and output a greater number of classifications. The network consists of two convolutional pooling layers, with one fully connected hidden layer. The features that are output by the hidden layer are used for classification by a Softmax classifier. The architecture of the system is shown in Figure 1.

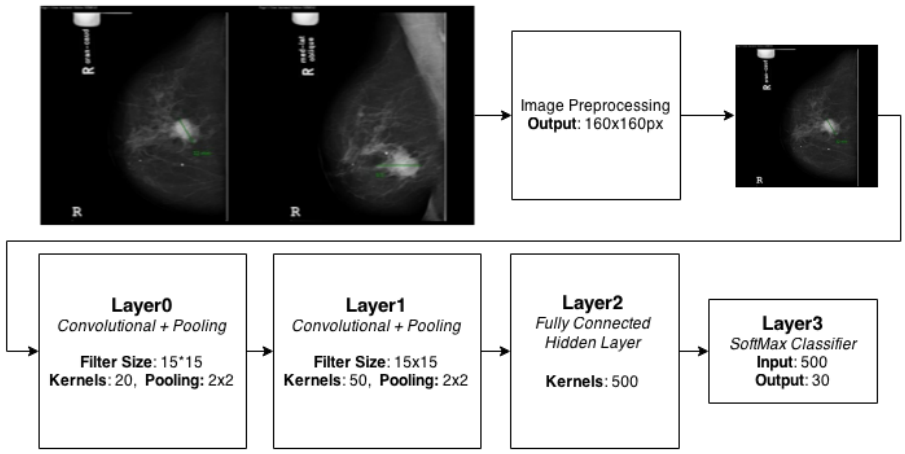


**Fig. 1.** The architecture of CNN used for the experiments

The specifications of the convolutional-pooling layers are detailed in Table 1.

**Table 1.** Details of Convolutional Pooling Layers

| Hyperparameter | Layer0 | Layer1 |
|---|---|---|
| Number of Filters | 20 | 50 |
| Size of Filters | 15x15px | 15x15px |
| Max Pooling | 2x2 | 2x2 |
| Stride | 1 | 1 |

[5] `http://deeplearning.net/tutorial/lenet.html`

Other hyperparameters for the CNN are detailed in Table 2.

**Table 2.** Other details for CNN

| Hyperparameter | Value |
|---|---|
| Number of Units in Fully Connected Layer | 500 |
| Batch Size | 20 |

As mentioned earlier the CNN requires a great deal of computational resource to run. It took approximately 3.5 hours to train a single epoch for each model, while training two models simultaneously on the CPU of a powerful system[6]. However, the models were not able to converge before the submission deadline. As such the runs that we submitted were based on only partially converged models. The details of the four runs submitted are detailed in Table 3.

**Table 3.** Run-specific details of all submissions.

| Submission | Model | Learning Rate | Training Epochs |
|---|---|---|---|
| `sf_run_1` | Softmax | 0.05 | 1000 |
| `sf_run_2` | CNN | 0.005 | 47 |
| `sf_run_3` | CNN | 0.005 | 55 |
| `sf_run_4` | CNN | 0.007 | 46 |
| `sf_run_5` | Softmax | 0.05 | 1000 |
| `sf_run_6` | CNN | 0.005 | 59 |

## 4 Validation Results

The validation error for all runs is displayed in Table 4.

**Table 4.** Validation error of all submissions.

| Submission | Validation Error |
|---|---|
| `sf_run_1` | 0.0% |
| `sf_run_2` | 13.85% |
| `sf_run_3` | 8.94% |
| `sf_run_4` | 10.53% |
| `sf_run_5` | 5.3% |
| `sf_run_6` | 6.53% |

[6] Azure Standard A4 VM: 8-core 2.1GHz CPU, 14GB RAM

### 4.1 Softmax Classification

The first Softmax model, trained for 1000 epochs, produced a 0% validation error. This was interpreted as being the result of severe over-fitting to the supplied training data. Despite the fact that this classification scheme was essentially a baseline to evaluate the performance of CNN-extracted representations over the raw data, it was thought prudent to perform second run, with less training and hopefully less overfitting, in order to see the results of a more general model. Thus, for `sf_run_5` we submitted the results of training the same model for only 155 epochs, this resulted in a 5.3% error rate on the validation set.

### 4.2 Convolutional Neural Networks

The validation errors displayed in Table 4 for the CNN runs (`sf_run_2,3,4,6`) demonstrated a clear correlation between the number of epochs they were trained for and increasing performance (decreasing validation error). This is demonstrated visually in Figure 2.

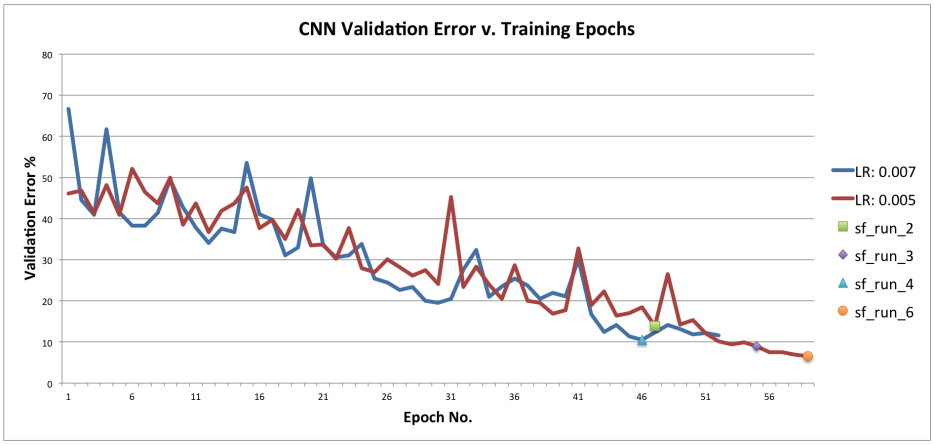


**Fig. 2.** The validation error of the models decreases as they approach convergence. The scatter points correspond to the epoch and validation error for each of the four test submissions.

## 5 Test Results

The test results for the six runs as supplied by ImageCLEF are displayed in Table 5. The CNNs demonstrated improved performance over the Softmax classification and their accuracy approximately corresponded to the amount of training that was performed.

**Table 5.** Test results as supplied by ImageCLEF.

| Submission | Correctly Classified |
|---|---|
| `sf_run_1` | 37.56% |
| `sf_run_2` | 43.62% |
| `sf_run_3` | 45.63% |
| `sf_run_4` | 44.34% |
| `sf_run_5` | 37.56% |
| `sf_run_6` | 45.00% |

### 5.1 Softmax Classification

The test accuracy for both runs of the Softmax classifier were 37.56%. This indicates that despite cutting short the training for `sf_run_5` compared to `sf_run_1`, both models had effectively the same representation of the data when it came to classifying the test data.

### 5.2 Convolutional Neural Networks

Compared to the validation results for the CNNs, the improvements with regard to the number of training epochs are not so clear-cut. For the model trained with learning rate of 0.005% there is a clear improvement between the test submitted at epoch 47 and the test submitted at epoch 55. However, the test accuracy decreased in the run submitted at epoch 59.

## 6 Analysis of Results

### 6.1 Validation vs. Test variance

An examination of the validation error and test results in Tables 4 and 5 is very illuminating. Clearly the 70/30 training/validation method we applied was inappropriate in this case, as demonstrated by the significant variance between the validation and test performance. While it is possible that the test set was substantially different to the training set, it's more likely the 30% chosen for validation was not fully representative of the data. Given that the training data was not evenly distributed in all classes it is likely that the models overfit the data corresponding to the more common classes and that the validation set was heavily skewed towards the common classes. However, we still believe that CNNs are suitable for this task despite the evidence of overfitting in this case. Techniques for overcoming this issue are discussed in Section 7.

Having already pointed out the tremendous computational demands required by the CNNs, more robust validation procedures such as 10-fold cross-validation are clearly not feasible with the system employed in this test. That said, it may be possible to perform this kind of validation on a simpler model such as Softmax, in order to discover a more indicative training/validation split. Another option

would be to take a more manual approach to splitting the sets, ensuring that all classes are evenly represented in the validation set.

It's worth noting that the models used for testing were only trained on the 70% training split. In future, we can expect better results by retraining the best model (based on some validation metric) on the entire dataset.

### 6.2 CNN Training

As alluded to out earlier, although the CNNs did not converge during training, they may have already begun to overfit the training data with the result that the test performance actually decreased for the model at epoch 59 compared to the model at epoch 55. However, this is not entirely certain, as it is also possible that the model at epoch 59 was a better fit for the validation data (table 4) at that point, but simultaneously a worse fit for the test data. Had the models been able to train for longer, we may have had a clearer indication of their true performance.

### 6.3 CNN-Learnt Features

The CNNs were able to extract improved representations from raw data without the requirement for domain knowledge. This is an important result both for this task and for MIR generally as it suggests that there is potential in using CNN or other deep learning strategies as a 'black box', whereby we will be able to achieve excellent machine learning performance without the need of expert-designed feature extraction or domain knowledge.

We would have liked to train the network further, but need to prematurely halt the system for the purposes of submission. We believe that additional training would yield a better result.

## 7 Perspectives for Future Work

We believe that that these results can be significantly improved upon by making use of a variety of techniques. Primarily we would want to explore training the CNNs using GPUs, as this will allow us to expand our hyperparamter and architecture search. Rectified Linear Units (ReLUs), as opposed the Tanh units used in our network are also known to improve training performance [16, 17].

Although this network is very capable of learning quality representations of the MNIST dataset, it is both less deep and less dense than networks used to achieve state-of-the-art results in more sophisticated tasks [8]. For instance, Krizhevsky et. al. [8] used a network with 2 convolutional-max pooling layers, 3 convolutional layers and 3 fully connected layers, all of which were more neuron-dense that ours, to achieve their result in ImageNET 2012. Improved training performance will allow us to implement a larger and deeper network along these lines.

Larger and deeper networks introduce issues with overfitting, but we believe this can be controlled using well-tried techniques such as dropout [8, 12, 18], data augmentation [8, 19] and unsupervised pretraining [20, 21].

Finally, the validation method we utilised for these experiments did not produce an accurate understanding of the performance of our systems. In approaching this task in future we would be careful to construct a more representative validation set or use the 2015 test data for validation.

**Acknowledgements**

This work was supported in part by a Microsoft Azure for Research grant, which provided the cloud infrastructure to conduct our experiments.

## References

1. M. Villegas, H. Mller, A. Gilbert, L. Piras, J. Wang, K. Mikolajczyk, A. G. S. de Herrera, S. Bromuri, M. A. Amin, M. K. Mohammed, B. Acar, S. Uskudarli, N. B. Marvasti, J. F. Aldana, and M. del Mar Roldn Garcia, General Overview of ImageCLEF at the CLEF 2015 Labs, Springer International Publishing, 2015.
2. A. Garcia Seco de Herrera, H. Mller, and S. Bromuri, Overview of the ImageCLEF 2015 medical classification task, in Working Notes of CLEF 2015 (Cross Language Evaluation Forum), 2015.
3. Y. Bengio, A. Courville, and P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 17981828, Aug. 2013.
4. Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature, vol. 521, no. 7553, pp. 436444, May 2015.
5. J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw., vol. 61, pp. 85117, Jan. 2015.
6. D. G. Lowe, Object recognition from local scale-invariant features, in Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, 1999, vol. 2, pp. 11501157 vol.2.
7. A. Kumar, J. Kim, W. Cai, M. Fulham, and D. Feng, Content-based medical image retrieval: a survey of applications to multidimensional and multimodality data, J. Digit. Imaging, vol. 26, no. 6, pp. 10251039, Dec. 2013.
8. A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 10971105.
9. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, 2009, pp. 248255.
10. J. Kalpathy-Cramer, A. G. S. de Herrera, D. Demner-Fushman, S. Antani, S. Bedrick, and H. Mller, Evaluating performance of biomedical image retrieval systemsAn overview of the medical image retrieval task at ImageCLEF 20042013, Comput. Med. Imaging Graph., vol. 39, pp. 5561, 2015.
11. H. Mller, N. Michoux, and D. Bandon, A review of content-based image retrieval systems in medical applicationsclinical benefits and future directions, International journal of, 2004.

12. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, J. Mach. Learn. Res., vol. 15, no. 1, pp. 19291958, Jan. 2014.
13. Y. Bengio, Practical recommendations for gradient-based training of deep architectures, arXiv [cs.LG], 24-Jun-2012.
14. D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, A committee of neural networks for traffic sign classification, in Neural Networks (IJCNN), The 2011 International Joint Conference on, 2011, pp. 19181921.
15. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE, vol. 86, no. 11, pp. 22782324, Nov. 1998.
16. V. Nair and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807814.
17. A. L. Maas, A. Y. Hannun, and A. Y. Ng, Rectifier Nonlinearities Improve Neural Network Acoustic Models, W—&CP, vol. 28, 2013.
18. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv [cs.NE], 03-Jul-2012.
19. Classifying plankton with deep neural networks, Sander Dieleman. [Online]. Available: `http://benanne.github.io/2015/03/17/plankton.html`. [Accessed: 30-May-2015].
20. X. Glorot, A. Bordes, and Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, in Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 513520.
21. Y. Bar, I. Diamant, L. Wolf, and H. Greenspan, Deep learning with non-medical training used for chest pathology identification, in SPIE Medical Imaging, 2015, p. 94140V94140V7.