Deep Transfer Learning for Meteor Detection

Yuri Galindo[1], **Ana Carolina Lorena**[2]

[1]Instituto de Ciencia e Tecnologia - Universidade Federal de Sao Paulo

[2]Instituto Tecnologico de Aeronautica

Abstract. *In this paper, a pre-trained deep Convolutional Neural Network is applied to the problem of detecting meteors. Trained with limited data, the best model achieved an error rate of 0.04 and an F1 score of 0.94. Different approaches to perform transfer learning are tested, revealing that the choice of a proper pre-training dataset can provide better off-the-shelf features and lead to better results, and that the use of very deep representations for transfer learning does not worsen performance for Deep Residual Networks.*

1. Introduction

EXOSS Citizen Science[1] is a non profit organization that studies and monitors meteors in Brazil. It has more than 50 active stations equipped with CCTV cameras that monitor the night sky, automatically capturing the incidence of meteors and meteorites. Integrating data from multiple stations and using some specific software, the EXOSS project is capable of acquiring information about the trajectory, velocity, and probable landing site of the meteor. The cameras inevitably capture other events such as the passage of animals and planes, fireworks, and atmospheric events. These captures must be classified and filtered, a task which is usually accomplished manually. Part of the dataset used in this study was provided by researchers from UNIVAP (*Universidade do Vale do Paraíba*). The UNIVAP Observatory of Astronomy and Space Physics is one of the institutional EXOSS monitoring stations, and the used dataset contains 130 images (50 meteors and 80 non-meteors) that were captured by this station during the months of April and May, 2017. This dataset was further enriched with images from other EXOSS stations, resulting in 1000 meteor images and 660 non-meteor images.

Deep convolutional neural networks are capable of dealing with raw image data without any feature engineering, and can surpass human performance in image classification in some datasets [He et al. 2015]. With many parameters to be fitted, these networks can take full advantage of large quantities of data when aided by modern GPUs, leading to their current success in this realm. But this large quantity of parameters becomes a problem when there is limited access to data, as in our case of study, in which case the network is prone to overfitting the training data and achieving low predictive accuracy for new data.

In this paper we apply deep convolutional neural networks to the problem of detecting the presence of meteorites in the captured images. We mitigate the issues of training on a small dataset by making use of transfer learning. By training a deep network on a large dataset and transferring the learned features, we were able to fine tune the network on our data, achieving 96% accuracy and 0.94 F1 score. Different approaches to the use of pre-trained neural networks are investigated.

[1]http://press.exoss.org/

The main contributions of this work are: the application of state-of-the art techniques designed for large benchmarks in a novel small dataset; investigating the influence of using a dataset more visually related to ours for pre-training; and studying the influence of network depth for learning transferable features, experimenting with very deep representations.

This paper is structured as follows. Section 2 presents related work. Section 3 describes the materials and methods employed. Section 4 presents the methodology adopted in the experiments, whose results are shown and discussed in Section 5. Section 6 concludes this paper.

2. Related Work

Using the high level features learned by training a deep convolutional neural network (CNN) in a large dataset to substitute hand crafted features has seen considerable success. [Sharif Razavian et al. 2014] used a linear Support Vector Machine (SVM) classifier applied to the features learned by a deep CNN trained on ImageNet to other domains, achieving good predictive results. This approach was capable of outperforming problem specific architectures in tasks such as object classification, scene classification, recognition of bird species, recognition of flower types, and attribute detection. [Girshick et al. 2014] significantly surpassed contemporary approaches to object detection by using CNN features in the classification.

[Donahue et al. 2014] also experiments with features learned in ImageNet, and applied SVM classifiers to layer activations in order to outperform contemporary approaches in object and scene classification tasks. In another experiment, the learned features are compared to SURF, an interest point detector and descriptor that achieved success in object recognition. The experiment compared the prior solutions to a domain adaptation task using the original SURF provided features, to the same models with the CNN learned features as input. The use of CNN features improved the predictive performance of the models by more than 60%.

Other works investigated the impact of pre-training in databases more related to their goal, when compared to Imagenet. [Zhou et al. 2014] significantly outperformed [Donahue et al. 2014] and [Sharif Razavian et al. 2014] in the scene classification task by building a large scene-centric database, and using it for pre-training. They concluded that the features learned from training in an object-centric database are different from those learned in a scene-centric one. Our work differs by comparing performance of networks trained on two different object-centric datasets, of which the main difference is the presence of color and the complexity of the images, rather than the problem goal itself.

[Yosinski et al. 2014] and [Azizpour et al. 2016] studied the influence of network depth on feature transferability. The first work found that increasing network depth can worsen the performance on the target task, but showed this problem to be solvable by fine-tuning, in which case the added depth can improve performance. The second work concluded that increased network depth does not harm predictive performance. Both papers analyzed networks as deep as 14 layers, while in this work the shallowest model is comprised of 18 layers.

3. Tools and Methods

Convolutional neural networks work by grouping weights in kernels, also known as filters or feature detectors. A kernel is a small matrix, comprised of weights for a neighbourhood of pixels. Each filter is applied to the whole image in a process called convolution, resulting in an image in which each pixel value is the result of the sum of the pixel values in its neighbourhood from the original image, weighted by the kernel. The results of this filter go through an activation function, resulting in another image called an activation map, that indicates how much the feature detected by the filter was present in that point. Each map is interpreted as an image channel by the next kernels. The first filters learn general purpose features, such as the presence of edges and curves, while deeper filters will learn more abstract features such as recognizing a nose. The use of max-pooling reduces the size of activation maps between layers, and the small activation maps of the last layers can be used as high level features for classification algorithms such as SVM. In our case a 2 layer neural network with fully connected layers is used.

There is a considerable resemblance regarding the basic and most general features learned by CNNs [Yosinski et al. 2014]. Since these are the most important to our case, distant from the original Imagenet dataset, we considered that any architecture capable of performing well on Imagenet is a good candidate for weight transfer. We considered some architectures with good results on the object classification task, such as [Szegedy et al. 2015], capable of achieveing 7.89% top-5 single model error on ImageNet, [Simonyan and Zisserman 2014] that reports 7% error with the same metrics, and [He et al. 2015] and [He et al. 2016], that surpassed human performance with 5.71% and 4.49% of error rates respectively. The deep residual nets described in [He et al. 2016], also known as Resnets, were chosen due to their superior performance and their capability of sustaining very deep architectures.

Resnets differ from regular neural network architectures by implementing shortcut connections that perform identity mapping, as demonstrated in Figure 1. The input of a layer is summed to the input of a deeper layer, distant by two or more layers. This makes the preceding layers approximate a residual between the original input and an underlying mapping of the data, instead of directly approximating an underlying mapping. This strategy makes very deep networks viable, overcoming the problem of degradation that plain networks face when the number of layers is largely increased.

Instead of testing various learning rates for our models, we employed a learning rate finder, described in [Smith 2017]. The idea behind this method is to perform a gradual increase of the learning rate, while monitoring the accuracy. Our approach differed by plotting learning rate versus cross entropy loss, making it possible to identify for which learning rate the loss stops decreasing. The adequate order of magnitude for the learning rate is the largest for which the loss still decreases.

We further modify the learning rate by applying cyclic annealing, using a technique named Stochastic Gradient Descent with warm restarts, developed by [Loshchilov and Hutter 2016]. At the start of each cycle, the learning rate is restored to its original value, and is then reduced in a cosine shaped manner as the network works through the batches. The restarts allow the model to explore a multitude of local minima of losses, while the annealing facilitates the convergence to a local minimum. The length of the cycle is multiplied by a constant at each iteration, gradually focusing the model on

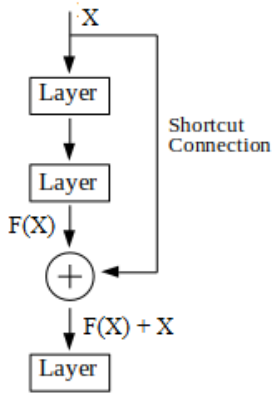

Figure 1. Use of shortcut connections in residual networks.

an optimal region.

As a weight optimization algorithm, we opted for ADAM [Kingma and Ba 2014]. ADAM works by calculating moving averages of the gradient and squared gradient, estimating the mean and uncentered variance of the objective function's gradient (the 1st moment and the 2nd raw moment, respectively). In [Kingma and Ba 2014] experiments, ADAM is shown to perform better than other optimization algorithms in the task of training CNNs in MNIST and CIFAR10 databases, beating established competitors such as SGD with momentum, AdaGrad, RMSProp and AdaDelta.

4. Experimental Methodology

The original dataset provided by the UNIVAP monitoring station contained 130 images, 80 non meteors and 50 meteors. Most of the non meteor examples from this station were planes, with some storm captures. Examples from this dataset are available in Figure 2. In order to attain more confidence in the results, the dataset was expanded by obtaining images available on the internet. Examples of meteors and atmospheric events were downloaded from the EXOSS website, and examples of miscellaneous objects were found in online archives of participants captures. The extended dataset is comprised of 1000 meteor images and 660 images of other objects in the night sky.

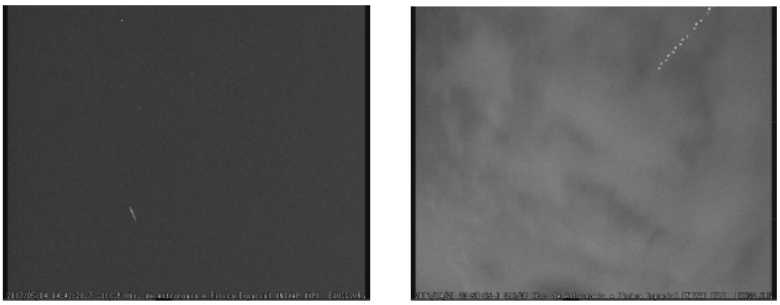

Figure 2. Meteor capture (left) and non meteor capture (right).

We investigated the effect of run-time augmentation on our models. Before being

fed to the network, each image is slightly zoomed, rotated in a random angle, and has a chance of being flipped. This augmentation is a way of artificially increasing the training set, and aids the model in learning angle invariance. We tested and compared models without data augmentation, with on side data augmentation (small angles and horizontal flip), and complete data augmentation. Since our experiments indicated that data augmentation worsens performance in this meteor classification case, models were trained without any data augmentation.

Most experiments involve fine tuning a pre-trained CNN, rather than learning the weights from scratch. The network is at first trained on a large dataset, in the case Imagenet or Fashion-MNIST. Fashion-MNIST [Xiao et al. 2017] is a benchmark dataset comprised of 70,000 images of various fashion products, intended to be a more complex substitute to the original MNIST digit recognition dataset. Similar to MNIST visually and in structure, the images have low resolution and the objects of interest are presented in grayscale against a black background. Since Fashion-MNIST is comprised of simpler black and white pictures, it is closer to our CCTV captures of the night sky. Imagenet, on the other hand, is a large visual object recognition dataset which presents colorful images with more clutter and detail.

The pre-training optimizes weights for the convolutional layers that are capable of extracting meaningful features from the raw data. The weights of the convolutional layers will be reused for the meteor detection, while the final and fully connected, non convolutional layers will be replaced with random weights to be trained in the meteor dataset. This process is the core of transfer learning, allowing us to use the knowledge learned in Imagenet or Fashion-MNIST for a different task. The features learned by the weights of the convolutional layers are used by the new fully connected layers, that learn how to use these features in the meteor detection context. We compare this approach to networks trained from scratch, to assess the usefulness of transfer learning.

After finding optimal weights for the fully connected layers on top of the learned features, we can also alter the convolutional weights, training the network as a whole. This process is called fine-tuning, and will find features that are more suited for the meteor dataset in lieu of Imagenet or Fashion-MNIST, possibly providing better results than using the features as they are. We kept the first 6 convolutional layers frozen because the very first layers tend to find general features, suitable for any dataset. We considered that a dataset as small as ours would not be capable of further optimizing these general features, trained on a very large dataset and capable of achieving very high accuracy in various domains.

Although the meteor images are in black and white, we used CNNs architectures designed for RGB images in the case of Imagenet. This allowed us to use the weights trained on Imagenet available on the PyTorch framework, that would require a few days to train from scratch if we decided to convert Imagenet into black and white before training. The conversion of the meteor images to RGB works by replicating the pixel intensity of one original channel to all three RGB channels. It is worth noting that since the convolution operation will output a single channel image for each filter, the only difference in the network architecture would be the dimension of the first layer filters, that would be reduced from depth three to depth one. [Hafemann et al. 2015] also pre-trained a network using a colored dataset for an application in a black and white context, but decided

instead to convert the pre-training images into black and white beforehand. They reported an insignificant performance gain.

The architectures used here are the same as described in [He et al. 2016], with a difference in the fully connected layers. While [He et al. 2016] uses an average pooling layer followed by a fully connected softmax layer, we apply another fully connected layer before the softmax layer. This layer is able to combine the convolutional features in another level of abstraction, useful when working without fine-tuning. The architectures used are available in 1. The filters are applied with stride 1 for layers that maintain the previous output size, and stride 2 for layers that reduce the output size.

Table 1. Network architectures

output size	18 layers	34 layers	50 layers	101 layers
112x112	64 7x7 filters, stride 2			
	3x3 max pooling, stride 2			
56x56	[64 3x3; 64 3x3] x2	[64 3x3; 64 3x3] x3	[64 1x1; 64 3x3; 256 1x1] x3	[64 1x1; 64 3x3; 256 1x1] x3
28x28	[128 3x3; 128 3x3] x2	[128 3x3; 128 3x3] x4	[128 1x1; 128 3x3; 512 1x1] x4	[128 1x1; 128 3x3; 512 1x1] x4
14x14	[256 3x3; 256 3x3] x2	[256 3x3; 256 3x3] x6	[256 1x1; 256 3x3; 1024 1x1] x6	[256 1x1; 256 3x3; 1024 1x1] x23
7x7	[512 3x3; 512 3x3] x2	[512 3x3; 512 3x3] x3	[512 1x1; 512 3x3; 2048 1x1] x3	[512 1x1; 512 3x3; 2048 1x1] x3
1	average pooling			
	fully connected layer, 1024 neurons			
	fully connected layer, 512 neurons			
	softmax layer			

In order to assess model accuracy, precision and recall, we used a complete 10-fold stratified cross validation. The data was divided into 10 folds, each maintaining the original proportion of meteors to non meteors. Afterwards, each fold is used for testing while the other nine folds are used for training the model. We compare model performance based on accuracy, averaged over the 10 runs.

5. Experimental Results

We compared the performance of different approaches in our meteor detection problem, investigating the efficacy of various methods for our particular case. These results can give us an indicative of how these techniques behave in a particular setting with few total images, validating previous studies on larger datasets. All experiments were performed on a NVIDIA Quadro P4000 GPU, equipped with 8GB memory and 1792 parallel workers.

Dropout was used in the fully connected layers, with a 50% dropout rate on the first fully connected layer and 25% on the second. In the Imagenet experiments, the learning rate finder procedure suggested a learning rate of 0.03 for training the fully connected layers, and a learning rate of 0.001 for fine-tuning. For training from scratch on the me-

teor dataset, a learning rate of 0.0002 was used, and a learning rate of 0.01 was used for training in Fashion-MNIST.

5.1. Data Augmentation

Run-time data augmentation worsened the results in our experiments, as shown in Table 2. This indicates that the transformations can be incompatible to the data, producing unrealistic trajectories that worsened classification. The issue is more pronounced when complete flipping and rotation are applied, since the resulting trajectories are more distant from the originals. Fine tuning also failed in this particular case, whilst in the other cases (no augmentation and on side augmentation) the results were improved after fine-tuning. The on side transformations preserve the trajectory better, and give results similar to using original non-augmented data. This first experiment was performed with deep networks of 18 layers pre-trained on the ImageNet dataset.

Table 2. Accuracy for different forms of run-time augmentation

	No Augmentation	On side Augmentation	Complete Augmentation
No fine tuning	89±2%	88±3%	88±3%
With fine tuning	**91±1%**	90±3%	87±3%

5.2. Network depth

We compared the accuracy achieved by networks with different depths (namely 18, 34, 50 and 101), all trained in ImageNet. As shown in Figure 3, we observed an improvement in the results when using deeper networks even when fine tuning is not performed. This is consistent with claims of [Azizpour et al. 2016] that the use of deeper features does not worsen representation, indicating that it holds for even very deep networks.

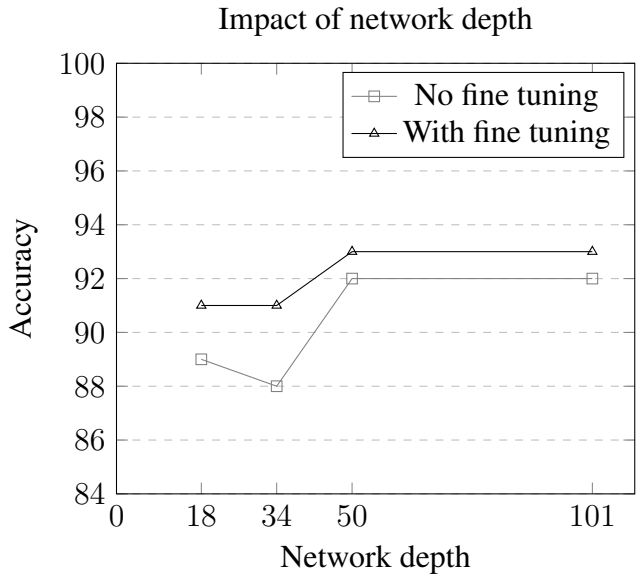

Figure 3. Comparison of classification accuracy for Resnets of different depths.

5.3. Pre-training

In this section we present comparative results of pre-training on two distinct datasets: Fashion-MNIST and ImageNet. We found that the use of Fashion-MNIST led to better off-the-shelf features, ultimately leading to better results, as seen on Table 3. The features learned in Fashion-MNIST outperformed the features learned by fine tuning a neural network previously trained in ImageNet, improving the F1 score of the final model from 91±2 to 94±1 . We compare here the deep network of 50 layers trained in ImageNet to a deep network of 18 layers trained on Fasion-MNIST and one trained from scratch. We opted for an 18 layer network for Fashion-MNIST because training on this dataset is computationally intensive, while in Imagenet the trained weights are available online. We also chose 18 layers for training from scratch due to overfitting in such a small dataset. The pre-training approaches were better than training a network from scratch. The obtained results highlight the importance of using initial networks which are pre-trained on datasets with more similar characteristics to those of the transferred domain.

Table 3. Accuracy between different pre-training datasets

	ImageNet	Fashion-MNIST	No pre-training
No fine tuning	92±2%	94±1%	-
With fine tuning	93±2%	**96±1%**	82±3%

5.4. Fine tuning

Fine tuning was shown to improve performance in our case study, as seen in Tables 2 and 3 and also in Table 4, being consistent with previous literature [Girshick et al. 2014]. Table 4 presents the impact of fine tuning in the deep networks of 18, 34, and 50 layers pre-trained in ImageNet, and in the deep network of 18 layers pre-trained on Fashion-MNIST. All models benefited from fine-tuning, reinforcing the usefulness of the approach.

Table 4. Impact of fine tuning across different models

	ImageNet 18	ImageNet 34	Imagenet 50	Fashion-MNIST
No fine tuning	89±2%	88±3%	92±2%	94±1%
With fine tuning	91±1%	91±2%	93±2%	**96±1%**

6. Conclusion

In this paper, state-of-the-art techniques were applied in a novel image classification problem: the recognition of images containing meteor captures. Different deep learning approaches were explored and compared, such as the use of fine tuning, the choice of a different pre-training dataset, the depth of the network, and the use of data augmentation. Our investigations lead to results consistent with previous literature, despite the relative small size of our dataset compared to those commonly used in the literature.

By pre-training a deep network in a black and white object-centric dataset, more similar to our captures of the night sky, we successfully achieved a low error rate (4 %) on our study case. We now plan to deploy the obtained model on the UNIVAP monitoring

station, filtering meteor images for the specialists. It would be also interesting to employ incremental models which may take advantage of new validated images and increase further the predictive performance achieved.

References

Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., and Carlsson, S. (2016). Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1790–1802.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Hafemann, L. G., Oliveira, L. S., Cavalin, P. R., and Sabourin, R. (2015). Transfer learning between texture classification tasks using convolutional neural networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Loshchilov, I. and Hutter, F. (2016). SGDR: stochastic gradient descent with restarts. *arXiv preprint arXiv:1608.03983*.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495.