Application of Machine and Deep Learning Methods to the Analysis of IACTs Data

Alessandro Bruno, Antonio Pagliaro, and Valentina La Parola

Abstract The Imaging Atmospheric Cherenkov technique opened a previously inaccessible window for the study of astrophysical sources of radiation in the very high-energy regime (TeV) and is playing a significant role in the discovery and characterization of very high-energy gamma-ray emitters. However, the data collected by Imaging Atmospheric Cherenkov Telescopes (IACTs) are highly dominated, even for the most powerful sources, by the overwhelming background due to cosmic-ray nuclei and cosmic-ray electrons. For this reason, the analysis of IACTs data demands a highly efficient background rejection technique able to discriminate gamma-ray induced signal. On the other hand, the analysis of ring images produced by muons in an IACT provides a powerful and precise method to calibrate the overall optical throughput and monitor the telescope optical point-spread function. A robust muon tagger to collect large and highly pure samples of muon events is therefore required for calibration purposes. Gamma/hadron discrimination and muon tagging through Machine and Deep Learning techniques are the main topics of the present work.

Keywords Machine learning · Deep learning · Atmospheric Cherenkov telescopes · Muons · Gamma/hadron separation · Image analysis · Pattern recognition · Computer vision

A. Bruno · A. Pagliaro (✉) · V. La Parola
INAF IASF Palermo, via Ugo La Malfa 153, 90146 Palermo, Italy
e-mail: antonio.pagliaro@inaf.it
URL: http://www.iasf-palermo.inaf.it/

A. Bruno
e-mail: alessandro.bruno@inaf.it

V. La Parola
e-mail: valentina.laparola@inaf.it

I. Zelinka et al. (eds.), *Intelligent Astrophysics*, Emergence, Complexity and Computation 39, https://doi.org/10.1007/978-3-030-65867-0_5

1 Introduction

When high energy particles and photons enter the Earth's atmosphere, they initiate a chain reaction of particles known as an atmospheric cascade. Secondary relativistic charged particles in the cascade emit Cherenkov light.

Imaging Atmospheric Cherenkov Telescopes (IACTs) can detect and image the Cherenkov radiation, allowing the observation of gamma-rays from the ground. However, gamma-rays contribute only to a small fraction of the flux of cosmic rays. The difficulty in suppressing the vast number of cosmic ray background events is one of the aspects that limit the sensitivity of IACTs. In order to detect gamma-ray sources an IACT analysis method must be able to perform an efficient background rejection, that is to separate the gamma-ray induced signal from the much more prevalent background of hadron induced showers, through the identification of shape features in the image. Luckily, the images generally contain sufficient information to separate the gamma-ray signal from the dominant cosmic rays background and reconstruct the arrival direction and energy of the primary.

Electromagnetic showers are characterized by an elliptically shaped shower image whose major axis is directed towards the source. If the primary particle is a cosmic ray, a hadronic shower develops. Although such hadronic showers often have electromagnetic sub-shower components as well, they lead to a typically more irregular shape. Analysis of IACT images relies on the extraction of relevant features from the camera pixel data. Whether those features are a vector of parameters representing the image, such as the image moments, or the full photo-electron intensity count in each pixel, in a Deep Learning approach they are automatically chosen by the network.

In general, Deep Learning concerns the application of complex Artificial Neural Networks to hierarchical learning tasks. For computer vision, Convolutional Neural Networks were designed specifically to perform image recognition tasks. We aim to develop and test several Convolutional Neural Networks Deep Learning architecture to determine the effectiveness of Deep Learning solutions over gamma/hadron separation.

High energy muons generated by air showers can be detected via their ring signature. In this paper, muon tagging is achieved by means of Machine Learning based on the extraction of relevant parameters from the data. We propose a feature set for muon tagging in order to automatically identify them.

2 The Simulations

The production of Cherenkov light in a shower induced either by a photon or by a particle in the atmosphere is a stochastic process. The shape, intensity and dimension of the observed Cherenkov pool depend on several factors, and most of them cannot be known with enough precision. The (unknown) height of the first interaction in the atmosphere, for example, introduces a large spread in the dimension of the light pool

at ground level, that reflects in large uncertainty in the evaluation of the energy of the primary particle. For this reason, the use of detailed simulation sets with different starting parameters and in different observing conditions is fundamental both for the calibration of the telescopes and for the analysis of the real data.

Throughout this work, all data sets used for training the different networks comprise air showers produced by high energy photons and cosmic rays generated by Monte Carlo simulations. These events are obtained by simulating the interaction of muons, gamma-rays and protons with the atmosphere using the CORSIKA software tool [Heck, Pierog and Knapp, 2012, Astrophysics Source Code Library, ascl:1202.006], that tracks the particles through the atmosphere while they undergo reactions with the air nuclei. An appropriate ray-tracing code follows the Cherenkov light produced by each shower when it is focused by the telescope mirrors. Finally, the image of the shower as it is recorded by the IACT camera on the focal plane is produced. The final image also includes the contribution of the night sky background. Simulated data trigger the telescope sensors giving rise to images. The camera images we produce at the end of the simulations are of size 56×56 pixels. Each pixel value represents the signal intensity. We produced:

- Muons: a set of muon events with energy between 6 and 1 TeV. Our final sample consists of 1500 muons and 1500 non-muons (both photons and hadrons).
- Gamma-ray photons and hadrons: two sets of Gamma-ray photons and hadrons simulations: one with standard night sky background, the second one with high night sky background. Events are distributed according to a realistic spectrum (with index -2.49 for the photons and -2.72 for the hadrons) between 3 and 100 TeV. We simulated 5075 photons and 5075 hadron events for each set.

3 The Muon Case

The optical throughput of an IACT is calibrated analyzing the image produced by highly energetic muons. Muons induce Cherenkov light emission, which, if the muon impact point is close to the optical axis, is imaged onto the focal plane as an arc or a ring.

Muon tagging is achieved by means of Machine Learning based on the extraction, from the camera data, of relevant parameters such as the image moments, the full photo-electron intensity count, the fullness and others. This task is not very computationally demanding, while the choice of the best parameters is crucial.

3.1 Image Cleaning Method

Muon tagging is applied to images cleaned from the Night Sky Background (Fig. 1). The cleaning maintains the basic shape of the signal and cancels isolated pixels. Our

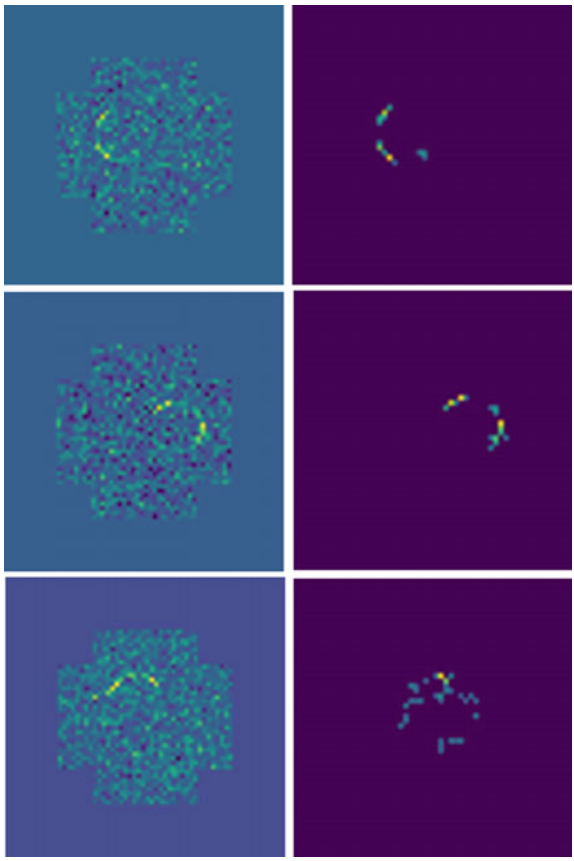


Fig. 1 Examples of muon events before and after the application of the cleaning method

method is based on a two-step cut algorithm. The first step cancels out all the pixel under a threshold τ. The threshold depends on the mean and standard deviation of the data and is computed as:

$$\tau = \overline{I} + k \cdot \sigma_I$$

where I average and root mean square are computed on the intensity of the image pixels, with no distinction between signal and Night Sky Background. Our best choice from trial and error for the value of k is $k = 2.5$.

The second step of the cleaning algorithm applies a 3x3 pixels window and selects only structures containing at least three pixels. Outliers are then removed: structures farther than eight pixels from the centre of mass are cancelled.

3.2 Choice of the Parameters

A set of sixteen discriminating parameters computed on the cleaned image has been chosen as input for our neural network. These are:

- The fractal dimension computed by means of the box-counting method (for a description of the box-counting method see [1]);
- The standard deviation of the fractal dimensions computed on different scales by means of wavelet methods as described in [2];
- The circularity computed as

$$Circularity = 4\pi\dot{(}Area/Perimeter^2)$$

  on the approximate polygon. We approximate a polygon from the largest connected structure by means of the Douglas-Peucker algorithm. Connected structures are selected defining feature connections with a 3x3 structuring element;
- The number of sides of the approximate polygon;
- The total intensity computed as the sum of the values of all the pixels;
- The first four Hu's moments computed on the approximate polygon (see [3]). Hu's moment invariants are a set of numbers calculated using central moments that are invariant to image transformations. The first six moments have been proved to be invariant to translation, scale, rotation, and reflection. If

$$\mu_{ij} = \sum_x \sum_y (x - \overline{x})^i (y - \overline{y})^j I(x, y)$$

  and we define normalized central moments as

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{(i+j)/2+1}}$$

  the first four Hu's moments are:

$$h_0 = \eta_{20} + \eta_{02}$$

$$h_1 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$h_2 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$h_3 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

  We found that Hu's moments higher than the fourth are not enough discriminating for our purpose;
- The total number of pixels over $3\sigma_I$, being σ_I the standard deviation of the uncleaned image;

- The average distance of the connected structures with three or more pixels from the centre of mass;
- The total number of non zero pixels in connected structures with two or more pixels;
- The maximum value of the radii of the connected structures with three or more pixels;
- The maximum distance of a connected structure from the centre of mass;
- The radius of the best fit circle computed on the largest connected structure;
- The fullness, defined as the number of non zero pixels inside a 2.5 pixels radius from the centre of the best fit circle computed on the largest connected structure.

The histograms of the values of the parameters for 1000 muon events and 1000 non-muon events are shown in Fig. 2. Histograms show how some of them are strongly discriminating, some are slightly.

3.3 Architecture and Results

The Machine Learning architecture used in our muon tagger is a linear stack of four layers, the input shape is made of the previously described sixteen parameters, hidden layers are made of 240 and 120 neurons, with a dropout of 0.2 after each stage, while the output stage is a single number between zero and one. Dropout is a technique widely used in machine learning methods to improve performances. It is a regularization technique applied to the output of some layers; some neurons are randomly dropped from the network during the training process. The dropout technique aims to prevent neurons of a network from relying on some other neurons during the training step.

We use the rectifier activation function on the first layers and the sigmoid function, that ensures our network output is between zero and one, in the output layer.

The algorithm has been applied on a mixed data set of muon and non-muon (both protons and gamma) events.

For the learning process we choose the following arguments: for the optimizer, we use the Nesterov Adam optimizer, which is essentially RMSprop with Nesterov momentum; for the loss function, we use binary cross-entropy.

We trained our neural network to identify muons. The learning process is made on a set of 1000 muons and 1000 non-muons events. After only 200 epochs the validation accuracy (on a 20% random samples of the data) is over 99%.

Our results for a test set of 1000 muons and 1000 non-muons events is as follows:

- True identification of non-muons: 96.1 %
- True identification of muons: 99.5 %

The selection power for muons is very high, while being not very computing demanding and pave the way for the development of high performance machine learning techniques run directly on the telescope camera server for the pre-selections of muon events.

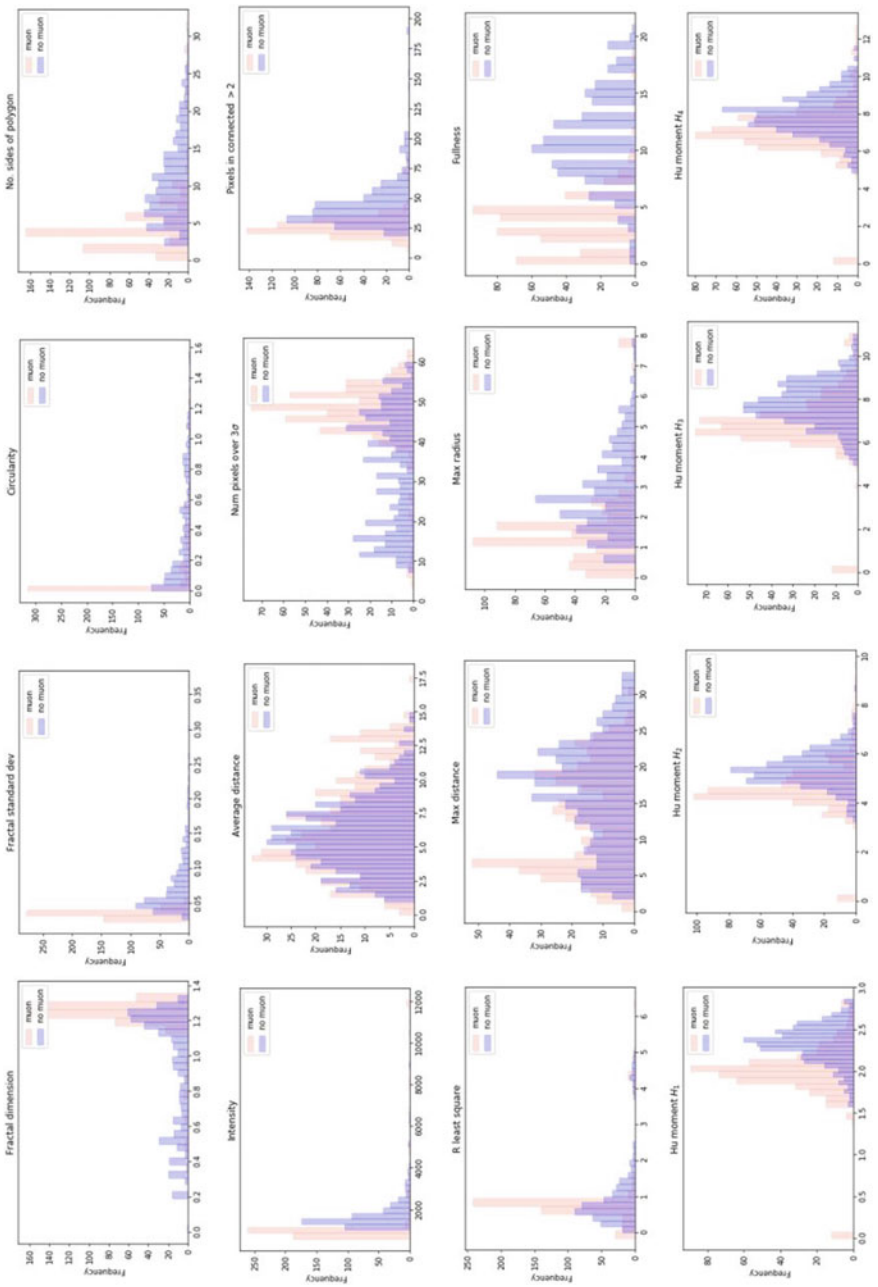


Fig. 2 Histograms for parameters used as inputs for the neural network. Clearer colour are muons, darker colour represents non-muons

3.4 Transfer Learning for the Muon Case

Transfer Learning is a Deep Learning approach which relies on pre-trained networks trained over a specific domain to be used over a new task domain (see [5]). A faster preliminary training step is needed rather than training networks from scratch. The latter scenario involves a large number of images to set-up a lot of parameters characterizing each CNNs.

The objective of this specific study is to analyse the power of the Transfer Learning paradigm to infer knowledge from a little number of images with constrained condition such as a strong background signal like the one present in the IACT domain. To this aim, we built a sample of 1500 muons plus 1500 non-muons images. The choice of 3000 as number of samples is not given by chance.

We want to find out a sort of trade-off between the number of images per class (muons and non-muons) and the test-accuracy of several Deep Learning architectures based on different principles and operations, trained over image classification tasks such as GoogLeNet, ResNet, SqueezeNet, MobileNet, putting in evidence how they affect the efficiency of the system.

Different pre-built Convolutional Neural Networks (CNNs), extensively described in the appendices, have been employed to find correlations between network depths, layers, hyper-parameters and performances over the detection topic. We conduct the study using a gradual approach, that is, first employing simpler networks made up of a lower number of layers and then, using networks with a higher number of layers. The list of the adopted networks follows:

- Flattened Deep Learning Architecture [6], which is one of the first basic and straightforward (Appendix A);
- GoogLeNet [8], based upon a graph with a large number of layers, characterized with higher depth than many state of the art approaches (Appendix D);
- ResNet-50 [7], made up of an ensemble of Residual Nets combining a graph (Appendix E).

In our work we customised CNNs pre-trained on ImageNet whose implementations are with Python packages such as TensorFlow Keras, PyTorch, and Torchvision. The latter is focused on image classification, semantic segmentation, and object detection.

To compare different performances among the above mentioned architectures we go through a multiple iterative refinement step to assess the best configurations for each given CNN. The step of parameters tuning relies upon trials with different epochs, number of images, number of classes for training.

3.5 *Experimental Results of Transfer Learning over the Muon Case*

First experimental results show a variety of performances which mostly depend on some parameters such as the number of images, the type of architecture and the number of epochs during training. Since we wanted to evaluate the performances of pre-built networks, we applied the Transfer Learning paradigm to transfer knowledge from a general domain to the specific IACT one. To this aim we focused our attention on CNNs pre-trained over ImageNet [11], an image database organized according to a semantic hierarchy called WordNet. The database counts in almost 14 millions of images. Several algorithms for object detection and image classification at large scale have been continuously tested and evaluated through the ImageNet Large Scale Visual Recognition Challenge (from 2010 up to 2017). In our work, we use four CNNs (GoogleNet, ResNet-50, MobileNet, SqueezeNet) pre-trained over ImageNet to be specialized over the muon case. As briefly mentioned before, the choice of Transfer Learning rather than training from scratch is required if we want to assess the power of Deep Learning methods using a strict constraint such as the number of images. In fewer words, we want to find out whether a connection can be found between the level of depth of the architectures we test and the level of knowledge inference abilities over a new application domain such as the IACT. In order to provide the artificial intelligence model with a sufficient number of images to set up parameters describing the application domain patterns and representations closely, we used Data Augmentation [12], a common practice in Transfer Learning.

We compared the test accuracy of the training process of the architecture: Flattened Network, GoogLeNet, ResNet-50 and SqueezeNet along with a growing number of epochs. We observe that different architecture-based CNNs have dissimilar performances. The experiments are conducted with a number of epochs up to 50 because the test accuracy function is observed not to increase its values, reaching out a sort of plateau. As we notice in Table 1, the first model we test is the one with the worst performance over the task of muon detection since it gets stuck with percentages around 60%. As we go along with a growing number of layer architectures, we notice better performances. In order, SqueezeNet, GoogleNet and ResNet-50 reach out test

Table 1 Validation accuracy values of CNN architectures for the muon case study

Validation accuracy of CNNs				
No. of epochs	Flattened network	GoogLeNet	ResNet-50	SqueezeNet
2	0.67	0.43	0.57	0.52
3	0.65	0.85	0.82	0.65
10	0.63	0.88	0.87	0.70
20	0.63	0.89	0.89	0.73
50	0.62	0.89	0.90	0.80

accuracy values equal or more significant than 80%. While SqueezeNet can achieve up to 80%, models with even a more-in-depth architecture, such as GoogleNet and ResNet-50 score top test accuracy values, 89% and 90% respectively. That said, all performances of GoogleNet and ResNet-50 being almost equal, ResNet-50 turns out to be preferable in terms of the computational burden. Validation accuracy is used as metrics to evaluate the performances of four CNN architectures and is defined as follows:

$$Validation\ Accuracy = \frac{Number\ of\ correct\ predictions\ on\ validation\ dataset}{Total\ number\ of\ predictions\ made}$$

4 The Gamma / Hadron Case

The capability to reduce the hadron background is one of the key aspects that determine the sensitivity of an IACT. Currently, the main data reduction and analysis software of the operating IACT facilities adopt methods based on Machine Learning algorithms. Although the performance of these methods has proven over the years to be rather robust and reliable, it may be overcome by new methods, in particular those based on Deep Learning.

To assess the performance of Deep Learning architectures we conducted several experimental sessions over raw data, whose Night Sky Background has not been suppressed.

Since in a Deep Learning approach the relevant parameters of the image are automatically chosen by the network, and the relevant shape features of the image are sorted out, Night Sky Background suppression is automatically achieved and no preliminary cleaning is requested.

4.1 Models

This section is devoted to the description of the Deep Learning architectures employed to test the Transfer Learning paradigm on the gamma/hadron separation from IACT data. In greater detail, we adopt the Convolutional Neural Networks (CNNs) [14] which are already pre-trained over the ImageNet repository. It is worth mentioning that CNNs have been widely adopted as one of the most accurate methods over tasks such as object recognition, suspicious region detection in biomedical imaging, speech recognition and semantic analysis. Each architecture needs to abide by some rules and constraints given by its own layers size, the number of layers, pooling, stride and hyper-parameters, which characterise the overall structure of the CNN

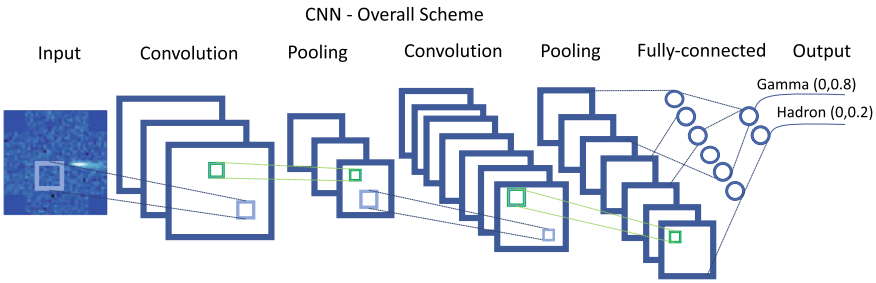
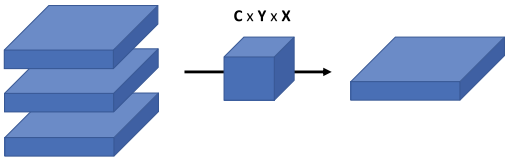
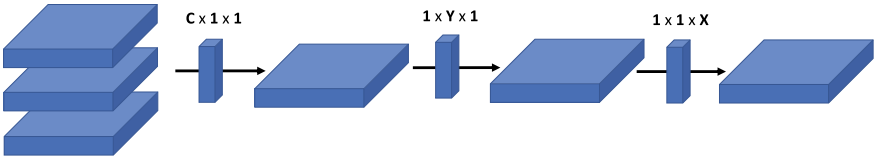


Fig. 3 The graphical representation of a basic CNN with convolutional and pooling layers

stack. Here we describe the most important detail of Flattened Network, GoogLeNet, ResNet-50, MobileNet, SqueezeNet. More specific descriptions can be found in the appendices. Figure 3 shows a graphical representation of a basic Convolutional Neural Network with convolutional, pooling and fully-connected layers, which are shared through the most of CNN architectures. Apart from the above-mentioned layers, a fundamental role is played by the so-called feature maps [15], that is, the output of filtering at each of the layers composing a CNN. A feature map is the result of the spatial filtering operation of input images with kernels such as the convolution and the pooling ones. The pooling layer and the stride parameter generally act to downsample the input image and extract features from the image itself, giving rise to a certain number of feature maps. The computational burden of the above steps is shared through the whole architecture and concerns mostly the size of kernels and pooling filters along with stride parameters. The role played by the size of filters is fundamental for the extraction of sized features in images. Filters with size 3×3, 5×5 or even 7×7 are conventional in CNN stacks. It is expected that the size of filters increases along with the dimensions of the images to inspect. The convolutional layer is meant to filter out some particular spatial features from input images. A spatial filter mask slides across the image using a sliding step parameter called stride. One can briefly state the output size of the filtering as well as the number and size of feature maps to be affected by two factors such as the kernel size and the stride value. The Flattened Network has been proposed by Jin et al. [6] to achieve the state-of-the-art results in terms of classification and recognition and to obtain a speed-up in the training time. All of that is achieved using a more lightweight architecture, that is, a reduced number of parameters to avoid redundancy of filters in the Convolutional Network. Jin et al. separated the conventional 3D convolution filters into three consecutive 1D filters. This step goes under the name of 3D filter separation under rank one. A graphic scheme is given in Fig. 4. The main idea behind it is all over the Convolution layer. The output of three consecutive 1D filterings gives an equivalent representation of the 3D filter if the rank is one. In greater details, Jin et al. chose a CNN model architecture as the one proposed by Srivastava and Salakhutdinov [13] with a smaller multilayer perceptron.



(a) 3D Convolution



(b) 1D Convolution over different directions

Fig. 4 In order, a graphical scheme of 3D convolution (upper row) and the corresponding 1D convolutions over different directions

5 Results

In this section, we give a detailed description of the experiments we ran with different CNN architectures and models. On the first instance, we detail the experimental sessions concerning the training process to draw some considerations about how diverse models suit the gamma/hadron separation. The dataset we use during our experimental trials consists of 10150 images with size 56×56 pixels and a standard Night Sky Background value. We split the whole dataset into two parts: the first 80% is used as a training set; the remaining images are used during the test trials. The training set is, in turn, split into two parts so that 20% of training images are used as a validation set. We highlight that in order to work with CNN architectures we need to abide by some requirements, such as the size of the input layer. Both test and training images go through resize transforms to make them CNN input layer compliant. In the mentioned architectures, SGD (Stochastic gradient descent) is adopted as an iterative learning algorithm to carry out the process of training over a dataset. Epochs and batch size are hyper-parameters which initialise SGD; the number of epochs controls the number of complete steps through the training set. The batch size handles the number of samples the training step goes through before the internal model parameters are updated.

We assess the accuracy of the CNN architectures using a different number of epochs (2, 3, 10, 20, 50). The batch size is experimentally fixed to 64. Dropout regularization parameter is set to 0.2. As a first step, we evaluate the performances of CNNs in the task of the gamma/hadron separation using the validation accuracy as described in Sect. 4. As it can be observed in Fig. 5, ResNet-50 achieves better performances than the other CNNs. Upon that, we decide to keep going on with our tests using ResNet-50 architecture and trying to assess the ability of knowledge inference over data apart from validation and training sets. As well as standard

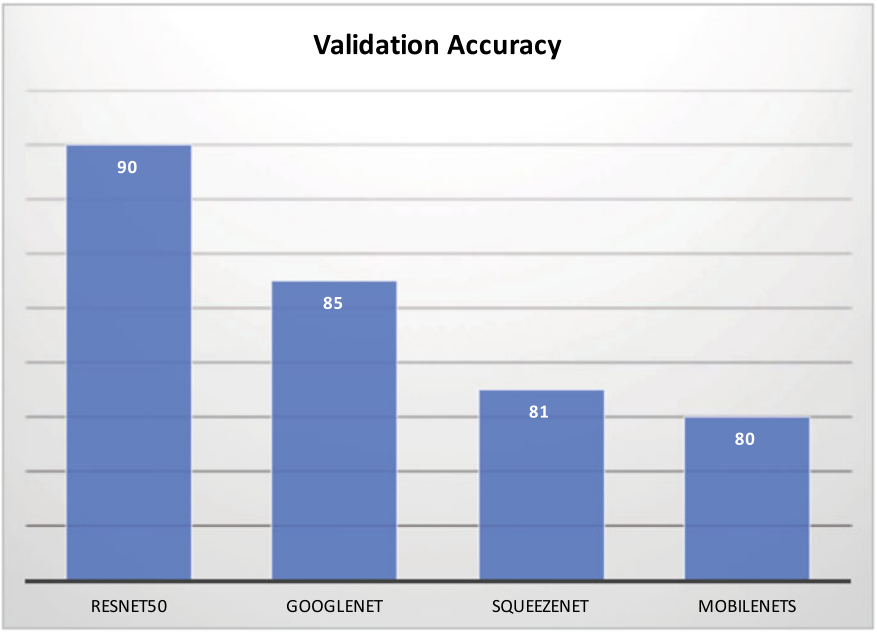


Fig. 5 Validation accuracy rates of four architectures are given with 50 epochs

Machine Learning metrics such as validation and test accuracy, we report results using the Quality Factor as defined down below:

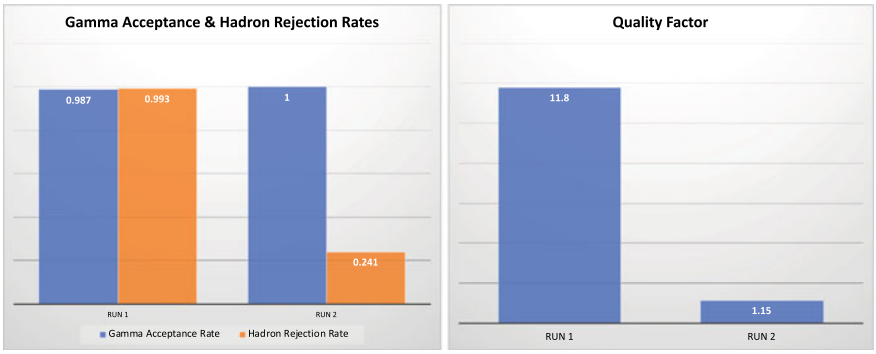
$$Q = \frac{\epsilon_\gamma}{\sqrt{\epsilon_{bkg}}}$$

From the equation above, ϵ_γ is the γ rate while ϵ_{bkg} represents the hadron acceptance rate. The γ acceptance rate is defined as the correctly classified γ events out of the total number of γ events. The hadron acceptance rate is defined as the ratio of proton events which behave like γ events after the γ-hadron classification. On the other side, the hadron rejection rate is the number of hadron events which have been correctly classified out of the total number of hadron events. The larger Quality Factor, the better gamma/hadron discrimination capabilities of the method.

We set up two runs with different Night Sky Background levels, described as it follows:

- Run no. 1 consists of 10150 images with standard Night Sky Background value (the same Night Sky Background as in the training set);
- Run no. 2 consists of 10150 images with one and a half times the Night Sky Background value in Run 1.

The network we fine-tuned over the gamma/hadron application domain appears to be Night Sky Background sensitive (see Fig. 6). Zooming in on Fig. 6a, b, our trained



(a) (b)

Fig. 6 Gamma Acceptance and Hadron Rejection rates are given for different experimental runs (a) and their corresponding Quality Factor values. The experiments have been conducted with the ResNet-50 model which is shown to perform better during the training step as shown in Fig. 11

model achieves respectively 98.7% and 99.3% of gamma and hadron acceptance rates on the run 1 while, as long as we analyse experimental trials with higher Night Sky Background we observe decreases in both rates coefficients (gamma and hadron acceptance rates). The corresponding Quality Factor values shown in Fig. 6a directly derived from gamma and hadron acceptance rates show the performance of ResNet-

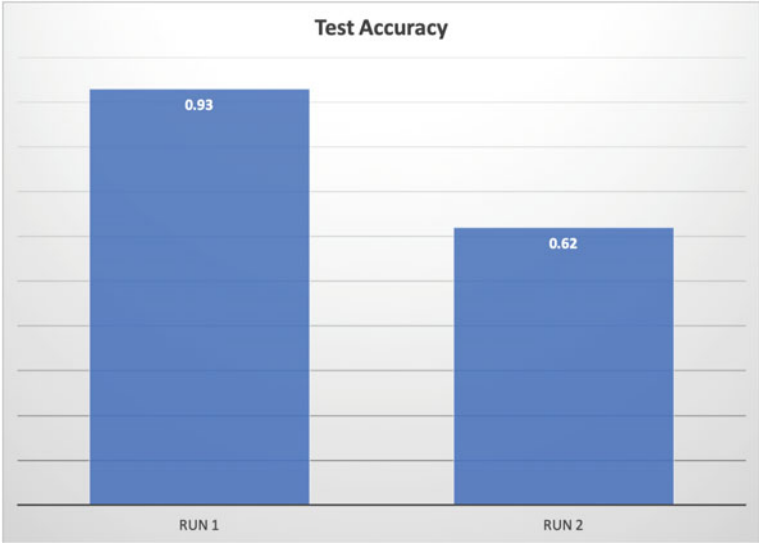


Fig. 7 Test accuracy rates of ResNet-50 model fine-tuned on gamma/hadron images for the gamma/hadron separation task. The test accuracy is defined as $Test\ Accuracy = \frac{Number\ of\ correct\ predictions\ on\ test\ dataset}{Total\ number\ of\ predictions\ made}$

50 equal respectively to 11.80 and 1.15 on run 1 and run 2. We want to highlight that ResNet-50 performances over run 1 simulation data is highly competitive with many state-of-the-art techniques as described in [19]. Because of our experimental results, Deep Learning architectures might resent from Night Sky Background values. From a pure detection viewpoint, looking at Fig. 7 higher values of test accuracy are shown to be over run no. 1 while results on run no. 2 dramatically decrease (the model is not able to make enough correct predictions of hadrons). Much more efforts would be necessary to clearly disambiguate the role played by the training set over the performances of the model on the test runs.

6 Conclusions

We have studied the application of different Machine and Deep Learning techniques to IACT images, with the aim of assessing their effectiveness in discriminating the images produced by muons, gamma-rays, and protons, respectively.

The muon case is relatively simple, since muon images have a well defined either circular or arc-shaped shape. This case was studied using a Machine Learning architecture, consisting of a linear stack of four layers. We selected a set of 16 image parameters, and we find that this approach is highly effective, reaching a 99.5% of correct identification over the muon sample, and 96.1% over the non-muon sample. We have also tested the effectiveness of a Transfer Learning approach over the domain of muon images. We find that the best results are achieved using the architecture with the highest level of depth. However Machine Learning is still more efficient.

The gamma/hadron discrimination case is more complicated: the difference between the two domains is vaguer whereas showers produced by hadrons may have an electromagnetic component, while showers produced by photons, especially by the low energy ones, or falling very close to the telescope axis may have a more irregular shape. In this case our study shows that an approach based on Transfer Learning may be effective in the discrimination, achieving a Quality Factor of ~ 12, highly competitive with many state-of-the-art techniques as described in [19]. Considering that we have trained the network only on a limited case sample, we expect that a more efficient training, including, e.g., images with different levels of Night Sky Background, off-axis photons, and/or a spectral distribution optimized for the training, may improve this result.

Appendices: Deep Learning architectures

A. Flattened Network

The Flattened Network [6] consists of a consecutive sequence of one-dimensional filters across all directions. The architecture of the network is simple and lightweight, and we use it in order to test whether such a simple network is affected by overfitting using a limited number of images. Models affected by overfitting usually make predictions that fit the data at hand perfectly, but are not able to generalize knowledge from larger datasets. This generally happens when the system does not discriminate information from bias or background noise embedded with data.

Each output channel requires a filter $W \in \mathbb{R}^{C \times X \times Y}$ described as:

$$F_f(x, y) = I * W_f = \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} I(c, x - x', y - y') W_f(c, x', y')$$

where f is an index of output channel, $I \in \mathbb{R}^{C \times N \times M \times F}$ is the input map, N and M are the spatial dimensions of the input. We assume the stride parameter to be one.

A rule of thumb to accelerate multi-dimensional convolution is to apply filter separation. To accomplish filter separation some constraints need to be considered. Under unit rank of the filter W_f, the unit rank filter $\widehat{W}_f$ can be separated into cross-products of three one-dimensional filters as follows:

$$\widehat{W}_f = \alpha_f \times \beta_f \times \gamma_f$$

It is necessary to highlight that separability of filters is a strong condition. The rank of filter W_f is usually larger than one in practice. The problem mentioned above might affect the performance of the network over classification tasks. In our work we used Flattened Networks abiding by the condition that one or more convolutional layers are converted to a sequence of 1-dimensional convolutions.

B. MobileNets

MobileNets [9] represent a class of efficient models based on streamlined architectures employing depth-wise separable convolutions to set-up lightweight deep neural networks. As an innovative solution Howard et al. [9] introduced two different global hyper-parameters as a trade-off between latency and accuracy of the network. MobileNets have different application domains, even though their ideal destination is to allow developers and computer scientists for testing and training CNNs over mobile devices and embedded vision applications. Depthwise separable filters represent the base on which MobileNets are built. Depthwise separable convolutions

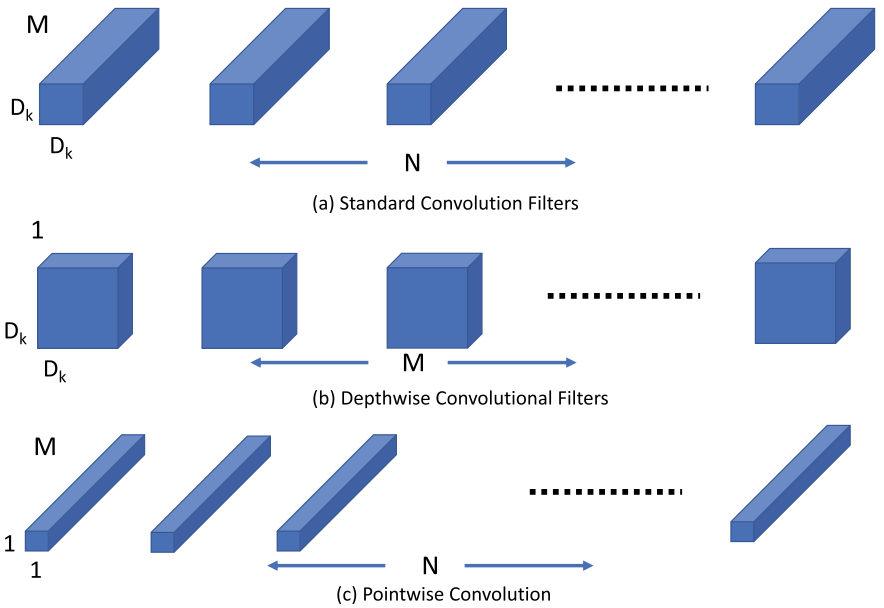


Fig. 8 MobileNets are based on Depthwise Convolutional Filters (**b**) and Pointwise Convolution (**c**), which make a noticeable parameter reduction over the architectures based on Standard Convolution Filters (**a**)

make use of factorization of convolutions into a depthwise and a 1×1 pointwise convolution. A standard convolution layer both filters and combines the input into a new set of outputs. The depthwise separable convolution splits this into two layers, the first one is devoted to filtering inputs while the second one is for the combination of inputs into a new series of outputs. The innovation of the proposed scheme in MobileNets is depicted as in Fig. 8 where standard Convolutional Filters are substituted with Depthwise Convolutional Filters and Pointwise Convolution allowing for a reduction of a great number of the architecture parameters. By zooming in Fig. 8 we notice that standard convolutions carry out a computational burden of:

$$D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f$$

where M represents the number of input channels, N is the number of output channels, $D_k \cdot D_k$ represents the size of the kernel while $D_f \cdot D_f$ is the feature map size. Depthwise convolution with one filter per input channel turns out to have a cost of:

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f$$

Depthwise convolution filters input channels but it does not combine the input into a new set of outputs. For this reason another layer is needed to combine the results of

Depthwise Convolutions filtering. That is accomplished with a linear combination of Depthwise Convolution using a 1×1 convolution. The latter one is called depthwise separable convolution whose computational cost is as it follows:

$$D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot M \cdot D_f \cdot D_f$$

As described in [9], a reduction in computation is achieved expressing convolution as a two-step process of filtering. By adopting 3×3 Depthwise separable convolutions, MobileNet is able to achieve up to 9 times less the computation than standard convolution. In our work we are interested in assessing the performance of MobileNets in the topic of IACT. For a more-in-depth description of the overall architecture the reader is remanded to [9].

C. SqueezeNet

SqueezeNet [10] architecture aims to leverage the reduction of parameters to deliver proper levels of accuracy in classification tasks with a short latency time as well as MobileNets. As well as in MobileNets, Iandola et al. [10] aim to identify a model that has fewer parameters in such a way to carry out experiments with more efficient training, to have less overhead time in client-server Deep Learning-based applications and to rely upon available architecture in embedded deployment. Iandola et al. [10] focused their efforts on the so-called model compression which has recently arisen around the objective of compressing existing CNN models in a lossy way. Denton et al. [17] applied SVD (Singular Value Decomposition) to pre-trained CNN models, Han et al. [18] used a pruning algorithm over networks to compress model dimensions. When it was introduced, SqueezeNet represented an innovation in the field of model compression for CNN architecture because of the introduction of the so-called fire module out of which the architecture itself is built.

In greater detail, Iandola et al.[10] proposed SqueezeNet by leveraging three strategies. The first strategy consists of replacing 3×3 filters with 1×1 filters (1×1 convolution filters have nine times fewer parameters than 3×3 convolution filters). The second strategy is to decrease the number of input channels to 3×3 filters. The third strategy consists of apply downsampling late in CNN to achieve larger feature maps as convolution layer output across the most of layers in the network. The first two strategies concern mainly decreases of parameters in CNN architectures, and the third one is focused on accuracy maximization with fewer parameters. A $Fire$ module consists of a squeeze convolution layer sized 1×1 which feeds into an expand layer that is a combination of 1×1 and 3×3 convolution layers. In Fig. 9 a simplified scheme of this module is given. Furthermore, the module can be tuned up using three different hyper-parameters. In our work, we adopt the architecture of SqueezeNet with a standalone convolution layer, followed by a series of 8 $Fire$ modules and a final convolution layer. Max pooling is performed along with the

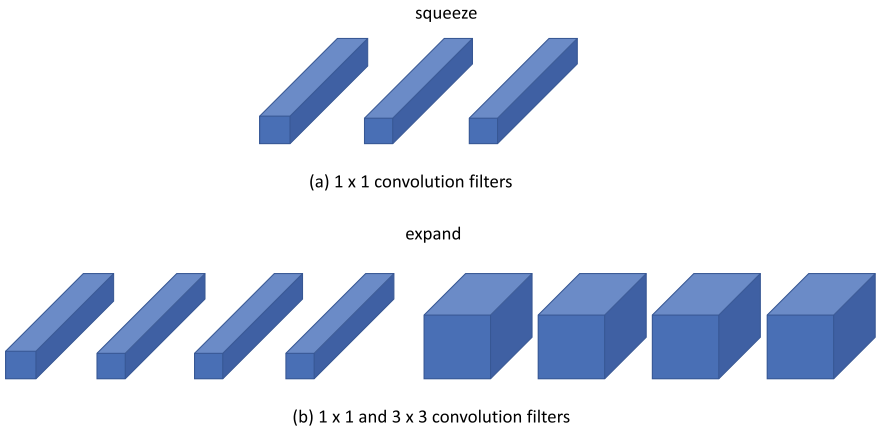


Fig. 9 SqueezeNet CNN architecture allows for a huge reduction of parameters, which is mainly based on the employment of squeeze (**a**) and expand (**b**) steps. The usage of 1×1 convolution filters make the CNN architecture more lightweight

network while the stride parameter is set to 2. The reader who is interested in further and more detailed description is remanded to the reference paper [10].

D. GoogLeNet

GoogLeNet is crafted to be an efficient deep neural network for computer vision tasks and its performances over the context of classification have been widely assessed [8]. Szegedy et al.[8] proposed an architecture whose main hallmark is the improved utilization of computing resources in the network itself. The main idea behind the GoogLeNet architecture is to find out how sparse structures in a convolutional network can be approximated by dense components. The authors of GoogLeNet engineered a network using a layer-by-layer approach where high statistic correlation values of the last layer are used to group visual features (boundaries, edges, contours, motifs) in clusters. These clusters give rise to units of the next layer and, at the same time, are connected to the previous layer. $Inception$ module is described in Fig. 10. The current version of $Inception$ includes layers with filters sized 1×1, 3×3, 5×5. Furthermore, max-pooling filters are added in the architecture as successful elements in state-of-the-art CNNs. Each unit from the earlier layers corresponds to a region of the input image. As depicted in Fig. 10, the visual information coming out of the previous layer is conveyed onto the next one through both convolution and pooling filters which are combined using a filter concatenation. $Inception$ modules are piled up on top of each other. The outputs of inception modules are statistically correlated with the corresponding layers of the network: features of higher levels are expected to decrease in spatial density when captured by higher layers. It is necessary

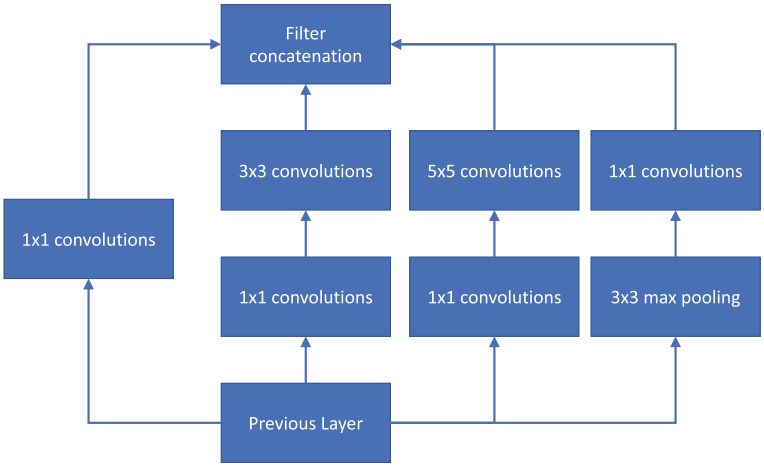


Fig. 10 The Inception module allows for multiple outputs coming out of the previous layer to be combined using a filter concatenation

to highlight that because of technical reasons, such as memory efficiency, *Inception* modules are added only at higher layers. The overall network counts in 22 layers when only layers with parameters are considered (no pooling layers are counted in). If we consider each block inside *Inception* modules the overall networks can count up to 100 layers (this number is affected by the infrastructure system set-up). Whoever interested in further insights on the architecture of GoogLeNet is remanded to the reference paper [8].

E. ResNet-50

He et al. [7] proposed Deep Residual Learning to generally ease the process of training of deep networks, which are likely to get through the problem of degradation. When the depth of networks increases accuracy gets saturated and then degrades rapidly. This kind of degradation is not caused by overfitting, and what sounds more surprisingly is that adding more layers makes the training accuracy even lower. He et al. [7] addressed the issue of training accuracy degradation with the introduction of Deep Learning Residual. Rather than considering Deep Learning models fitting a particular mapping between input and output, they approach the improvement of training accuracy from a different perspective. They let a stack of layers fit a residual mapping function. Denoting the original mapping function as $H(x)$, the Deep Residual Learning is based on the position of letting the layer stack fit the residual function defined as down below.

$$F(x) := H(x) - x$$

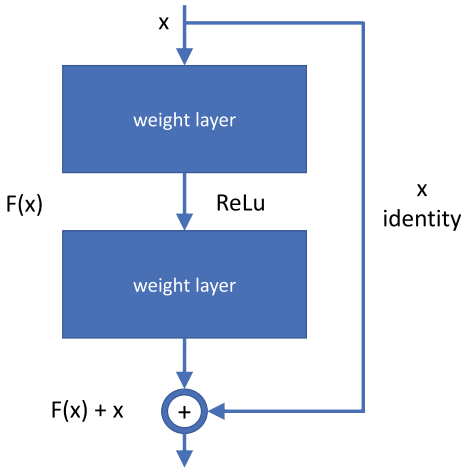


Fig. 11 A building block of Residual Net

Following this approach, the original mapping problem is proposed in a new form as $F(x) + x$. As a new formulation based on a simple summation, it can be realised using feedforward networks with shortcut connections as in Fig. 11. Shortcut connections simply add new layers with identity mapping. Their outputs are added to the outputs of the stack. Using shortcut connections is an excellent way to avoid new parameters because they are based on identity mapping. The intuition of Deep Residual Learning authors is that multiple nonlinear layers can asymptotically approximate complicated function representing the input of the training process. The reformulation of learning is meant to deal with the training accuracy degradation problem. He et al. named ResNet (Residual Network) after the Deep Residual Learning formulation. The standard version of ResNet is 34 parameter layer sized where shortcut connections turn the network structure into its residual counterpart. A more in-depth version of ResNet has also been proposed with a bottleneck building block whose each residual function F involves a number of three layers rather than two. In our work we conducted different experiments with ResNet-50, which counts a number of 50 parameter layers.

References

1. Li, J., Du, Q., Sun, C.: An improved box-counting method for image fractal dimension estimation. Pattern Recognit. **42**(11), 2460–2469 (2009)
2. Pagliaro, A., D'Anna, F, D'Alí Staiti, G.: A multiscale, lacunarity and neural network method for γ/h discrimination in extensive air showers. In: Proceedings of the 32nd International Cosmic Ray Conference (2011)
3. Huang, Z., Leng, J.: Analysis of Hu's moment invariants on image scaling and rotation. In: Proceedings of 2nd International Conference on Computer Engineering and Technology (2010)

4. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**, 195–197 (1981)
5. Weiss, K., Khoshgoftaar, T., M., Wang, D.: A survey of transfer learning. J. Big Data **3**(1), 9 (2016). SpringerOpen
6. Jin, J., Dundar, A., Culurciello, E.: Flattened convolutional neural networks for feedforward acceleration (2014). arXiv:1412.5474
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
9. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., and Wang, W., Weyand, T., and Andretto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017). arXiv:1704.04861
10. Iandola, F. N., Song, H., Moskewicz, M. W., Khalid, A., Dally, W. J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size (2016). arXiv:1602.07360
11. Russakovsky, O., Deng J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y (2015)
12. Wang, J., Perez, L.: The effectiveness of data augmentation in image classification using deep learning. In: Computer Vision and Pattern Recognition (2017)
13. Srivastava, N., Salakhutdinov, R.R.: Discriminative transfer learning with tree-based priors. In: Advances in Neural Information Processing Systems, pp. 2094–2102 (2013)
14. Agarwal, R., Diaz, O., Llad'o, X., Yap Moi, H., Mart'i, R.:Automatic mass detection in mammograms using deep convolutional neural networks. J. Med. Imaging **6**(3), 031409 (2019)
15. LeCun, Y., Boser, B., Denker, J.S. et al.: Handwritten digit recogntion with a back-propagation network. In: Advances in Neural Information Processing Systems (1990)
16. LeCun, Y., Bottou, L., Bengio, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
17. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: NIPS (2014)
18. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural networks. In: NIPS (2015)
19. Sharma, M., Nayak, J., Koul, M. K., Bose, S., Mitra, A.:Gamma/hadron segregation for a ground based imaging atmospheric Cherenkov telescope using machine learning methods: random Forest leads. Res. Astron. Astrophys. **14**(11), 1491 (2014)