Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Image orientation detection by ensembles of Stochastic CNNs

Alessandra Lumini [a], Loris Nanni [b,*], Luca Scattolaro [b], Gianluca Maguolo [b]

[a] DISI - University of Bologna, Campus di Cesena, Via dell'Università 50, 47521 Cesena (FC), Italy
[b] DEI, University of Padua, Via Gradenigo 6, 35131 Padova, Italy

ARTICLE INFO

Keywords:
Image orientation detection
Convolutional neural networks
Photo orientation
Deep learning
Transfer learning

ABSTRACT

In this paper we deal with the problem of accurately and automatically detecting the orientation of general images, for instance, of holiday snapshots. Detecting image orientation is an easy task for a human being but can be a long and tedious activity during processing and management of digital photos. Several attempts have been made in the design of systems for automated displaying images in their correct orientation, however, this is still an open problem. In this work we exploit the power of deep learning proposing a transfer learning approach that adjusts pre-trained convolutional neural networks to this classification task. We create ensembles of different Convolutional Neural Network models designed by randomly changing the activation functions in all the activation layers of a given network. Along with several known activation functions we also include the novel Soft Learnable activation function in the "random set". Our resulting ensembles have been extensively evaluated on more than 45,000 images taken from four different public datasets, showing a remarkable performance improvement with respect to other state-of-the-art approaches. All the source code used for this work is freely available at https://github.com/LorisNanni/.

1. Introduction

The advances of digital imaging and the large number of digital cameras, smartphones and other devices have led to a significant increase in the number of photographs captured by people. Since the camera is not always horizontal during the capturing process, the resulting photos often need rotation correction in order to be displayed in the correct orientation, i.e. the orientation in which the scene originally occurred. Most digital cameras have a built-in orientation sensor and allow to record the orientation of the camera during shot in the EXIF metadata of the image, however this field is not consistently managed and updated by several image processing application and image formats. Thus, automatic detection of canonical image orientation is an important task for several applications e.g. automated creation of digital albums, digitization of analogic photos, computer vision applications which require input images in the upright orientation. Even if any angle of rotation is possible, rotations by multiple of 90° are the most common, and are also straightforward to correct once detected. Moreover, among the possible four orientations 0° rotation is the most likely, since most of the photos acquired by a digital camera are landscapes, sometimes the acquisition device is rotated by 90° or 270°, and very seldom rotated by 180°.

Even if human beings can solve this problem with high accuracy (about 98% with a resolution of 256 × 348 according to (Ciocca et al., 2015)), automated image orientation detection is a challenging task due to the great variance of image content. Older approaches (i.e. Ciocca et al., 2015; Lumini & Nanni, 2006) were based on low-level features of an image, but recent advances in deep learning have changed the entire landscape over the past few years, thanks to their ability to deal with most image processing tasks very effectively and efficiently.

Convolutional Neural Networks (CNNs) are one of the most effective tools in modern computer vision. Most state-of-the-art performances on tasks like image classification, image segmentation and object recognition are currently achieved using CNNs. However, experiments show that the performance of CNNs can be improved by training a large number of models and averaging their results. Such a set of neural networks is called an ensemble. The idea behind averaging the results of multiple classifiers is that multiple opinions are better than only one opinion. Besides, in this way every network can specialize on a specific part of the data. Finally, a method to fuse the decision of the different networks is required.

Ensembles are very old tools in machine learning and date back to much earlier than the rise of modern deep learning (Tukey, 1977). The difference between the performance of single models and ensembles has been widely investigated in the literature and there is strong evidence that ensembles outperform single models (Kuncheva & Whitaker, 2003). The key to improve the generalization ability of an ensemble is to have a diverse and accurate set of classifiers which are at least somewhat uncorrelated. Ensemble research is a very active area and

* Corresponding author.
E-mail addresses: alessandra.lumini@unibo.it (A. Lumini), loris.nanni@unipd.it (L. Nanni).

https://doi.org/10.1016/j.mlwa.2021.100090
Received 26 December 2020; Received in revised form 7 June 2021; Accepted 21 June 2021
Available online 25 June 2021
2666-8270/© 2021 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

a lot of methods have been proposed to generate ensemble methods able to solve the existing trade-off between diversity and accuracy (Brown et al., 2005). The combination approaches in ensembles can be classified into two main categories: learned and not learned. In the former case, the rule for combining the results is trained using training or validation data. For example, the rule might be a weighted average of the single softmax output of a classifier. In the latter case, the choices might be the average of the softmax output or the class that represent the mode of the model predictions. A more complex learned model is, for example, the evolutionary learned method proposed in (Chandra & Yao, 2006) for the development of ensembles tackling the trade-off between diversity and accuracy. Anyway, the computational complexity of such methods makes them unsuitable to be coupled with deep networks.

The main drawback of ensemble methods is the fact that they require more computation than a single model and they have more memory requirements. The application of ensemble techniques to CNNs is more recent and follows the popularity of deep learning for computer vision problems: in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012, the winner was an ensemble of the well-known AlexNet. These ensembles were made by simple retraining the same architecture multiple times with different initializations and using the stochasticity of the training process. This approach has been proved (Fort et al., 2020) to be very effective: the trajectories of randomly initialized neural networks explore different modes in function space, which explains why deep ensembles trained with just random initializations work well in practice.

In general, there are many ways to create a diverse ensemble. The easiest one consists in using different models and architectures. The hypothesis is that every model will learn a specific feature of the training data. Then, one might train the same model on different training sets. This can be done by randomly splitting the training data into subsets, or by using different data augmentation techniques in every training (i.e. a different pre-processing for every network (Brahnam et al., 2020). Another method relies on the difference of the training processes, even when the models and the datasets are the same.

In this work we propose an image orientation detection algorithm based on deep ensembles based on a perturbation of the network architecture. We deal with the most common classification problem in four classes related to rotations by multiples of 90°. We train several convolutional neural networks (CNNs) to automatically determine the original image orientation. We create different CNN models by randomly changing the activation functions in all the activation layers of a given network. The set of activation functions used in this work includes several known activation functions which are listed in Section 3 and a novel Soft Learnable activation function here proposed. The use of different activation functions allows each network to learn different patterns while achieving good performances. In this way, we managed to create an ensemble of neural networks that outperforms every single network. The resulting ensembles are trained on an extensive dataset of more than 45,000 images and tested on four different datasets. The average classification accuracy of the system is more than 90%, which is a better performance with respect to the existing state-of-the-art approaches (Ciocca et al., 2015; Swami et al., 2017).

The rest of the paper is organized as follows. In Section 2 we discuss some related works. We outline the CNN architectures tested in this work in Section 3, along with some details about the training procedure. In Section 4 we present the datasets used for experiments and the testing protocol and we discuss our experimental results comparing them to prior work. The main contribution of this work consists of an approach obtaining improvement of performance with respect to published state-of-the-art methods on the task of image orientation classification. Besides, the source code will be freely available at https://github.com/LorisNanni/.

2. Related work

Most of the related works that deal with this problem restrict the classification to the main 4 standard orientations, while there are few works that aimed at regressing the exact orientation angle (Cao et al., 2016; Fischer et al., 2015; Olmschenk et al., 2017; Prince et al., 2019). We are mainly interested in the literature about the classification into the 4 canonical classes, since orientation regression is a slightly different task, as the angle of rotation is usually more limited, and hence outside the scope of this paper. This problem is a long-standing topic of research. Before the deep learning revolution, existing approaches were based on low level features, such as color histograms, edge direction histograms, and texture features (Vailaya et al., 2002), edge-based structural features and color moment features (Wang & Zhang, 2004), a set of natural image statistics collected from a multi-scale multi-orientation image decomposition (Lyu, 2005), Harris corner, phase symmetry, edge direction histogram (Lumini & Nanni, 2006), Local Binary Patterns (Ciocca et al., 2015), or domain specific visual cues, such as object recognition and the contextual information present in the image (i.e. text Solanki et al., 2004), faces, horizon (Fefilatyev et al., 2006)). The main drawback of such approaches is that handcrafted features are limited due to the semantic gap between low-level features and high level image semantics, while visual cues are domain specific and fail when dealing with general images including a wide variety of both indoor and outdoor photos. For these reasons, more recent studies have turned to convolutional neural networks to predict the orientation of an image. In Swami et al. (2017) a transfer learning approach based on a pretrained AlexNet architecture is proposed and trained on a balanced dataset (images were equally distributed among the 4 classes). In (Shima et al., 2017) a face orientation problem is handled using AlexNet as a feature extractor and SVM as trainable classifier. In Joshi and Guerzhoy (2018) the authors used a pretrained VGG16 network: their results include performance on a large consumer dataset (balanced by including all the possible rotations of each photo) and a visual analysis about how the CNN detects photo orientation. In Morra et al. (2019) the authors propose a method based on fine tuning a slightly modified version of VGG16: some fully connected layers are removed in order to make the network lighter, and the dropout and batch normalization layers are used differently from the original VGG16. Another interesting aspect of that work is that a further class, labeled undefined, is introduced to deal with images whose original orientation is not available/sure. In Prince et al. (2019) a two-stage method is introduced: the first step of the method uses deep learning (specifically a fine-tuned AlexNet) to determine the canonical orientation, and in the second step fuzzy logic is used to determine a more precise angle of the image.

3. Proposed method

The method proposed in this work is based on a variant of known CNN architectures (i.e. ResNet, GoogleNet, MobileNet, Densenet), based on the stochastic substitution of activation layers (Nanni et al., 2020). Activation functions play a crucial role in discriminative capabilities of the deep neural networks and the design of new "static" or "dynamic" activation functions is an active area of research. In this work, we propose an ensemble of models obtained by using a mixture of "static" and "dynamic" activation functions, which are stochastically selected at each activation layer to substitute the standard ReLU (Rectified Linear Unit) function.

As proposed in Nanni et al. (2020), we randomly substitute all the ReLU activations in our starting model with a random activation function selected among a given pool of activations. Hence, multiple activation functions are used inside the same network at different levels, creating a stochastic version of the original model. We refer to this stand-alone network as *SingleStoc*. A SingleStoc model is obtained iterating along the layers of an input model (i.e. a ResNet) and replacing each activation layer of the input model by a different activation layer randomly selected among a pool of functions. The selection of the activation functions is not optimized to the specific classification task in order to avoid overfitting on the training set. On the other hand it is well known that one of the main problem of modern machine learning techniques is underspecification (D'Amour et al., 2020), i.e. when there are many distinct solutions that solve a given problem equivalently. Underspecification is a core idea in deep ensembles: in Fort et al. (2020) the authors claim that different solutions of the same model exhibit functional dissimilarity, hence, they are suitable to build ensembles. In this work we exploit dissimilarity given by diversifying activation functions used inside each model to design an ensemble.

The procedure used to generate each SingleStoc model is iterated ten times yielding a different network every time. Then the resulting networks are fused together by the sum rule used to create an ensemble of classifiers. We refer to this ensemble as *FusStoc10*. The sum rule, used for the inference of the ensemble, consists in summing all the output probability vectors of the networks in the ensemble and choosing the class whose sum of the probabilities is the highest. The idea behind the method is that all these networks are similar enough to the base CNN model to have similar or superior performances, but, at the same time, they are different enough to bring new useful information in the ensemble.

3.1. Activation functions

We shall now list all the activation functions contained in our pool. ReLU (Glorot et al., 2011), Leaky ReLU (Maas et al., 2013), Exponential Linear Unit (ELU) (Clevert et al., 2016), Parametric ReLU (PReLU), S-Shaped ReLU (Jin et al., 2016), Adaptive Piecewise Linear Unit (Agostinelli et al., 2015), Mexican Linear Unit (Maguolo et al., 2019) and Gaussian Linear Unit are used, as proposed in Nanni et al. (2020). Some of these functions depend on a parameter called *maxInput*. This parameter controls the shape of the activation and it is recommended to be set at the maximum value of the input images, i.e: 255. However, we tested some of those functions both with *maxInput* equal to 1 and some of them with *maxInput* equal to 255. Among the functions already used in Nanni et al. (2020), we assume that ReLU, leaky ReLU, ELU and PReLU are already well known and we are going to explain S-Shaped ReLU, Adaptive Piecewise Linear Unit, Mexican Linear Unit and Gaussian Linear Unit.

S-Shaped Linear Unit (SReLU) is defined as

$$y_i = f\left(x_i\right) = \begin{cases} t^l + a^l(x_i - t^l), & x_i < t^l \\ x_i, & t^l \le x_i \le t^r \\ t^r + a^r(x_i - t^r), & x_i > t^r \end{cases} \tag{1}$$

where t^l, t^r, a^l, a^r are learnable real numbers. The name of the activation comes from the fact that its shape recalls the one of the letter S. In our implementation, the parameters a^l, a^r are respectively initialized to 0 and 1, in order to make SReLU equal to ReLU in its first steps. The parameters t^l, t^r are respectively initialized to 0 and $maxInput$.

Adaptive Piecewise Linear Unit (APLU) is defined as

$$y_i = \text{ReLU}\left(x_i\right) + \sum_{c=1}^{n} a_c \max(0, -x_i + b_c) \tag{2}$$

where a_c, b_c are real numbers that are different for every channel of the input. This is a piecewise linear function whose slopes and points of non-differentiability are learnable. In order to ensure a stable training, its creators suggest to use a 0.001 L^2-penalty.

Mexican Linear Unit (MeLU) is defined as

$$y_i = MeLU\left(x_i\right) = PReLU\left(x_i\right) + \sum_{j=1}^{k-1} c_j \phi_{a_j,\lambda_j}(x_i) \tag{3}$$

where

$$\phi_{a,\lambda}(x) = \max\left(\lambda - |x-a|, 0\right) \tag{4}$$

is a function that has its maximum in a and linearly decreases in both directions until it reaches zero. It has the shape of a Mexican hat, which gives the name to the activation. The hyperparameter k is the number of learnable parameters in every channel. These learnable parameters are the c_j and the one of PReLU. a_j and λ_j determine the height and the position of the function $\phi_{a,\lambda}$ and they are chosen recursively so that, for every function $\phi_{a,\lambda}$, the next two functions have a height which is one half of the original one and have a support which is exactly one half of the original support. We refer to the original paper for further details.

Gaussian Linear Unit (GaLU) is defined in the same way as MeLU, with the difference that the basis function is defined as

$$\phi^{a,\lambda}(x) = \max\left(\lambda \cdot maxInput - |x - a \cdot maxInput|, 0\right) + \min(|x - a \cdot maxInput - 2\lambda \cdot maxInput| - \lambda \cdot maxInput, 0) \tag{5}$$

GaLU receives its name from the fact that its basis function recalls the shape of a Gaussian. The parameter *maxInput* is set at the maximum value of the input images, i.e: 255.

We also included in our pool new activation functions that we shall now list and describe. These functions are Parametric Deformable Exponential Linear Unit, Swish, Mish, Soft Root Sign and Soft Learnable.

Parametric Deformable Exponential Linear Unit (PDELU) was introduced in Cheng et al. (2020). It is defined as:

$$y_i = f\left(x_i\right) = \begin{cases} x_i, & x_i > 0 \\ a_i \cdot \left([1 + (1-t)x_i]_+^{\frac{1}{1-t}} - 1 \right), & x_i \le 0 \end{cases} \tag{6}$$

It was designed to have mean equal to zero, which allows faster training. Its output belongs to the range $[-a, +\infty)$.

Swish is an activation function introduced in Ramachandran et al. (2018), it is a smooth, non-monotonic activation function which was automatically learnt through reinforcement learning. It is defined as:

$$y = f(x) = x \cdot sigmoid(\beta x) = \frac{x}{1 + e^{-\beta x}} \tag{7}$$

where β is a parameter. We included two versions of Swish in this paper: one with β learnable and one with β fixed.

Mish was introduced in Misra (2019): it is a smooth and non-monotonic activation function defined as:

$$y = f(x) = x \cdot tanh(softplus(\alpha x)) = x \cdot tanh(ln(1 + e^{\alpha x})) \tag{8}$$

In our experiments the parameter α is also learnable.

Soft Root Sign (SRS) was introduced in Zhou et al. (2020). It is defined as:

$$y = f(x) = \frac{x}{\frac{x}{\alpha} + e^{-\frac{x}{\beta}}} \tag{9}$$

where α and β are a pair of trainable non-negative parameters. SRS is not monotone and it has a region where it is negative. According to the authors, if the distribution of the input x is a standard normal, SRS has zero mean, which helps to speed up training.

A new activation function is proposed in this work, named Soft Learnable activation function, which is defined as:

$$y = f(x) = \begin{cases} x, & x > 0 \\ \alpha \cdot ln\left(\frac{1+e^{\beta x}}{2}\right), & x \le 0 \end{cases} \tag{10}$$

where α is a trainable parameter and β is optionally trainable. They are both constrained to be positive. We used two versions of this function in the paper, one with β fixed (SoftLearnable) and one with β learnable (SoftLearnable2). The function takes its name from the Soft Plus function (i.e: $f(x) = \log(e^x + 1)$), that inspired this one, and from its learnable parameters. Soft Learnable is similar to the Soft Plus function for negative values, except from the two parameters and from the fact that, even when $\alpha = \beta = 1$, Soft Learnable is ln(2) lower than Soft Plus. Notice that in this way Soft Learnable is continuous in zero. For

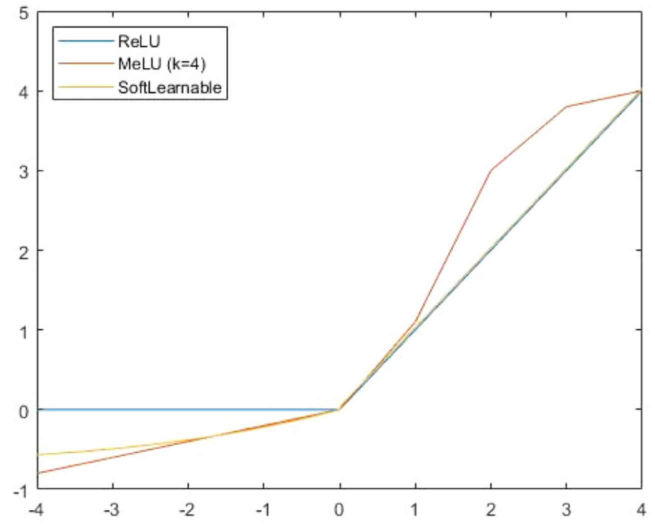

Fig. 1. Plot of some activation functions: ReLU, MeLu and SoftLearnable.

positive values Soft Learnable is equal to the identity function. In this way, we created a new activation that does not saturate on the right, that is bounded from below and that controls its slope on the left. A plot of SoftLearnable compared to ReLu and MeLU is reported in Fig. 1.

3.2. *CNN architectures*

CNNs are deep neural networks designed for image classification; CNNs are feed-forward multilayer networks designed to work similarly to the human brain in visually perceiving the world. CNNs may include several types of layers:

- Convolutional layers, whose aim is to substitute conventional handcrafted feature extractors by convolving input to filters that preserve the spatial relationship between pixels.
- Activation layers, which are used to introduce nonlinearity to the system;
- Pool layers reduce the dimensionality of each feature map but retain the most important information with the purpose of making the input representations smaller and reduce the risk of overfitting.
- Fully-connected layers connect every neuron in the previous layer to every neuron on the next layer. They are used at the end of the network for classification purposes. They are usually coupled to a SoftMax or a Sigmoid function to transform a vector of arbitrary real-valued scores to a vector of values between zero and one.

In this work we use four different pre-trained models modified in order to fit the new classification problem (i.e. changing the last fully-connected layer to fit the number of classes, without freezing the weights of the previous layers) and “fine-tuned” with the training set of the current problem. In our experiments, we test and combine the following different CNN architectures available in the MATLAB Deep Learning Toolbox; all the models, which are pre-trained on a large dataset of objects (the ImageNet database) are “fine-tuned” on the current problem:

- GoogleNet (Szegedy et al., 2015) is a 22 layers deep network including the so-called “Inception” module, i.e. a subnetwork consisting of parallel convolutional filters whose outputs are concatenated, with the aim of making the network deeper, but strongly reducing the number of parameters. This network has been selected since it is a good compromise among complexity and performance.
- ResNet50 (He et al., 2016) was the winner of ILSVRC 2015. ResNet introduces a new kind of layer, named residual, including a “network-in-network” architecture. This model, here used in the middle-size version of 50 layers, is one of the best performing and most popular architecture used for image classification.
- DenseNet (Huang et al., 2017) is a very deep network having 201 layers. DenseNet in an evolution of ResNet which connects each layer to every other layer in a feed-forward fashion, thus increasing the number of connections. This architecture has been selected for its higher performance, even if it is a quite heavy architecture.
- MobileNet (Sandler et al., 2018) a light-weight architecture that uses depth-wise separable convolutions to perform well in mobile devices. The architecture has been selected for its low computational requirements.

In all the experiments aimed at comparing the activation functions, the ResNet50 architecture is used. Each model obtained by substituting all the ReLU activation layers by the same activation function (fixed, not stochastically selected) is named by the name of the activation, i.e MeLU is the model where all the ReLU layers have been replaced by MeLU layers. When the activation function is ReLU the model is the original architecture (ResNet50, or others).

4. Datasets and evaluation

This section describes the datasets used in this paper and explains how the data was prepared for training and testing. To assess the performance of the proposed system we have used the testing protocol proposed by (Swami et al., 2017) consisting in a cross dataset evaluation.

4.1. *Datasets*

The datasets used in our experiments are the following (summarized in Table 1):

- SUN397 (SUN) is a large scale database (Xiao et al., 2016) for scene recognition which contains 103,754 images in 397 different categories (each category having at least 100 images). For training purposes, we selected 45,000 images from this dataset, while leaving the remaining in the test set.
- INRIA Holidays (INRIA) dataset (Jegou et al., 2008) includes 1491 images from a very large variety of scene types (natural, man-made, water and fire effects, etc.). The dataset is divided into 500 image groups, each of which represents a distinct scene or object.
- MIT INDOOR (MIT) (Quattoni & Torralba, 2009) dataset contains 15620 indoor images in 67 categories (with at least 100 images in each category). We used only the recommended test set of 1340 images for testing.
- PascalVOC 2012 (PASCAL) (Everingham et al., 2010) is a dataset for object recognition including 11540 images. Each image contains a set of objects, out of 20 different classes. We used only the training and validation dataset of 6233 images for testing.
- HUMAN (https://zenodo.org/record/4758038#.YJ1aDqgzY2w). It is a human centric dataset created by the authors of this paper for training purposes, in order to improve the performance for orientation classification in presence of a human being. It contains images from several public datasets for skin segmentation: in total 5541 images representing one or more human beings.

All the images in the five datasets are in their correct orientation, even if some orientation-ambiguous images are present. Since all our CNN models require input images to be of size $224 \times 224 \times 3$, we simply resize all the images to the required dimensions (without padding).

Table 1
Datasets description.

Dataset	Short name	Ref.	#images	#training	#test	#classes
SUN397	SUN	Xiao et al. (2016)	103754	45000	58754	397
INRIA Holidays	INRIA	Jegou et al. (2008)	1491	0	1491	500
MIT INDOOR	MIT	Quattoni and Torralba (2009)	15620	0	1340	67
PascalVOC 2012	PASCAL	Everingham et al. (2010)	11540	0	6233	20
HUMAN	HUMAN	Self-collected	5541	5541	0	–

Table 2
Performance of ResNet50 varying the activation function: average accuracy and rank on 13 medical datasets for image classification.

Activation function	Average accuracy	Rank
ReLU	84.54	5
Leaky ReLu	80.30	16
ELU	82.79	15
PReLU	79.26	17
softLearnable2	83.45	11
softLearnable	83.17	14
PDELU	83.81	8
Learnable Mish	83.28	13
SRS	83.98	7
Swish Learnable	83.60	10
Swish	83.42	12
MeLU ($k = 8$)	85.26	1
MeLU ($k = 4$)	85.26	1
SReLU	85.15	3
APLU	84.35	6
Small GaLU	85.08	4
GaLU	83.73	9

For training purposes, we selected 45,000 images from the SUN dataset: each image has also been rotated by 90°, 180° and 270° degrees and labeled accordingly. Only for the last two experiments the HUMAN dataset has been added to the training set.

For testing purposes we used 58,754 images from the SUN dataset, 1340 images from the MIT dataset and 6233 images form the PASCAL dataset, which are the same subsets proposed in Swami et al. (2017). The authors of (Swami et al., 2017) selected a subset of INRIA by removing images with uncertain orientation. Since such subset was not available, we used the whole dataset (therefore the performance on INRIA is not comparable with Swami et al., 2017). All the test images have been transformed as in the training set (i.e. resizing and random rotation), with the difference that every image has been rotated only once, while in the training set for each original image we had four training images, i.e. one for each rotation.

4.2. Experiments

The first experiment aims to compare the activation functions included in our pool. To this aim we performed a standard image classification experiment using several medical datasets (please see (Nanni et al., 2020) for a detailed description of the datasets). In Table 2 the average accuracy and the rank are reported for each activation function ($maxInput = 255$ when appliable, i.e. for the last 6 approaches). For the sake of space, the whole table is included as supplemental material (see Table A.1), as medical image classification is not the focus of this work.

In this experiment the backbone CNN is ResNet50, and the other models are obtained by substituting all the ReLU activation layers by the same activation function. The best average accuracy on 13 datasets is obtained by MeLu.

The second experiment is aimed at evaluating the proposed approaches on the orientation classification problem. To this aim we fine-tuned several CNN of the SUN training set. In Table 3 we report the classification accuracy in each test dataset of both stand-alone methods and ensembles obtained starting from 2 different architectures: ResNet50 and GoogleNet.

The stand-alone models are the base ResNet50/GoogleNet with ReLU, the modified version using MeLU (the best activation function according to the previous experiment) and the SingleStoc stand-alone model where all the ReLU layers have been replaced by randomly selected activation functions. The ensembles are the sum rule between ReLU and Melu, the fusion of 10 SingleStoc models (FusStoc10) and the fusion of 10 base ReLU models having the same architecture (but different initialization before training).

In this experiment, all the models have been fine-tuned for image orientation detection according to the following parameters: batch size 50, learning rate 0.0001, max epoch 20. The only layer with a different learning rate is the last fully-connected, whose learning rate is 0.002 to better exploit transfer learning. Data augmentation includes random rescaling in both axes by two factors uniformly sampled in [1,2].

The following conclusions can be drawn from the reported results:

Table 4
Accuracy of different models (stand-alone or ensembles) on the 4 testing datasets.

Model	Variant	SUN	INRIA	MIT	PASCAL
ResNet50	ReLU	97.37	75.79	97.76	90.82
	FusStoc10	98.01	76.59	98.66	92.54
GoogleNet	ReLU	96.57	74.92	97.24	89.99
	FusStoc10	97.21	75.12	98.21	91.06
MobileNet	ReLU	96.91	74.92	97.91	90.45
	FusStoc10	97.73	76.73	98.58	91.93
DenseNet	ReLU	**98.08**	**76.86**	98.36	**92.67**
	SingleStoc	97.70	76.79	**99.10**	91.42

Table 3
Accuracy of ResNet50 and GoogleNet models varying the activation function on the 4 testing datasets.

Model	Variant	SUN	INRIA	MIT	PASCAL
ResNet50	ReLU	96.51	74.70	97.46	88.50
	MeLU	96.48	75.59	97.24	88.37
	SingleStoc	96.60	75.65	96.87	89.14
	ReLU+MeLU	96.74	75.43	97.54	89.06
	FusReLU10	96.77	75.05	97.39	88.80
	FusStoc10	**97.58**	**75.99**	**97.91**	**90.74**
GoogleNet	ReLU	96.21	75.52	97.01	88.71
	MeLU	96.26	75.52	97.10	88.77
	SingleStoc	95.32	73.71	95.82	86.70
	ReLU+MeLU	96.48	75.92	97.24	89.09
	FusReLU10	96.76	76.12	97.54	89.59
	FusStoc10	**97.11**	**76.32**	**98.06**	**90.00**

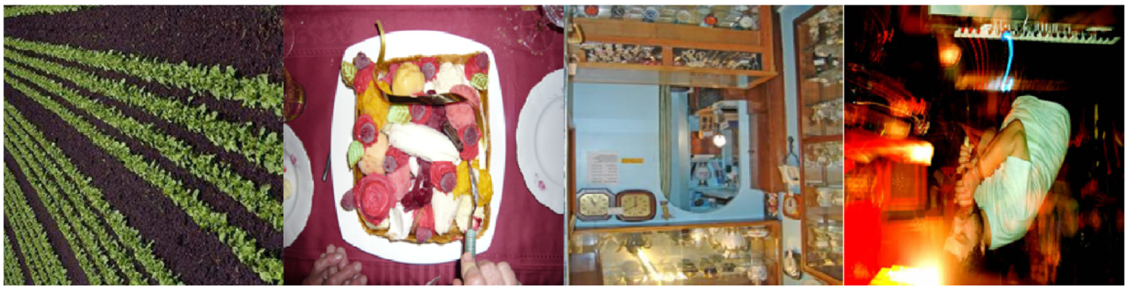

Fig. 2. Example of images non-correctly reoriented (one for each dataset).

Table 5
Accuracy on the MIT dataset and its variants of MobileNet models: stochastic, 2 base model with different initialization, ensemble of 10 stochastic.

	MIT	PIXEL	CON.	BRI.	BLUR
AVG(SingleStoc)	97.71 ± 0.42	77.90 ± 4.72	79.78 ± 2.72	76.09 ± 3.43	73.92 ± 1.79
MobileNet	97.91	76.79	86.19	85.00	81.41
MobileNet2	98.05	80.74	85.67	84.17	75.37
FusStoc10	98.58	85.90	86.79	82.16	80.97

- ReLU and MeLU obtain similar performance but their fusion (i.e. ReLU+MeLU) outperforms ReLU in all the four datasets.
- FusReLU10 (i.e. the ensemble of 10 models based on ReLU) has performance similar to a standalone ReLU, this means that running more times the same network does not permit to build a high performing ensemble.
- The ensemble named FusStoc10 clearly outperforms all the other methods for both the base architectures, therefore stochastic selection of activation functions is a feasible approach for building a high performing ensemble of networks.
- There is no tangible difference between ResNet50 and GoogleNet, they perform quite similar in this problem.

In two datasets (i.e. SUN and MIT), our ensemble of ResNet50 named FusStoc10 obtains a similar performance to that obtained by human beings and reported in Luo et al. (2003), which is around 98%.

As a further analysis on the networks involved in the ensembles, we have measured the error independence among the classifiers combined in FusReLU10 and FusStoc10 by evaluating Q-statistic (Kuncheva & Whitaker, 2003) among them. Q-statistic measures the homogeneity of a group of classifiers, which should be minimized to increase their diversity.

For ResNet50 the Q-statistic among the 10 FusReLU10 networks is 0.993, while it is lower in FusStoc10: 0.972. The same result can be observed for ensembles of GoogleNets: Q-statistic is 0.987 in FusReLU10 and 0.947 in FusStoc10.

This result means that stochastic selection of activation functions is a better way to increase the diversity of CNNs used to build an ensemble of networks.

We have carried out a visual analysis of the errors: in Fig. 2 some examples of images for which the system failed to predict the correct orientation are reported. In most cases, the correct orientation could be predicted by a human observer thanks to the presence of some details (a person, an object etc.). On the other hand, for the images in Fig. 3 the orientation is inherently more difficult to detect, also for humans, mainly due to a too restricted point of view.

While it is difficult to train the network to correctly recognize such cases of restricted point of view, we felt it was possible to teach the model to infer the correct orientation in presence of human being in the image. Since we noticed that the training set was mainly composed by images collected for scene recognition, thus not including human beings, we enlarged the training set adding the images from the HUMAN dataset. The training options have been set to: batch size 50, learning rate 0.0003, max epoch 20 (no data augmentation). We shuffle the training data before each training epoch.

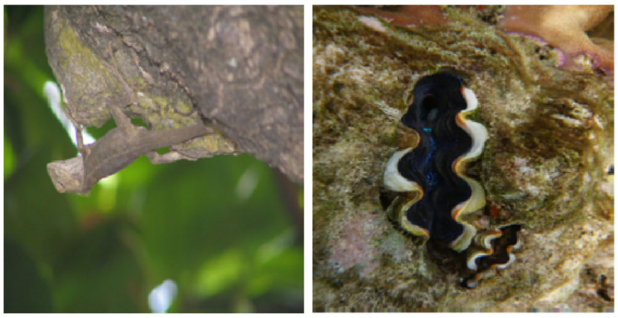

Fig. 3. Example of images non-correctly reoriented from the INRIA dataset.

In Table 4 we report the classification accuracy in each test dataset for the following stand-alone models and their related ensemble FusStoc10 (due to excessive training time the ensemble FusStoc10 of DenseNet is substituted by stand-alone SingleStoc). All the networks have been trained used the enlarged SUN+HUMAN training set.

From Table 4 it is interesting to note the improvement with respect to previous results in all the datasets. As the architectures are concerned, DenseNet gains very valuable result, but at the expense of a very long training time. MobileNet, which is a lightweight architecture, gains result only slightly lower than that of ResNet50 and similar to GoogleNet.

As a final experiment we perform a stress test to evaluate the robustness of the proposed ensemble under distribution shift, i.e. on some stress tests that have been proposed in the image classification robustness literature (D'Amour et al., 2020), which consist in applying synthetic but realistic corruptions to the test images, such as pixelation, high contrast, brightness and motion blur. Table 5 reports the performance of MobileNet FusStoch10 ensemble compared with the average accuracy (± standard deviation) of the 10 networks composing the ensemble and of two stand-alone MobileNets (MobileNet2 is the same network with a different initialization). The testing sets are, respectively, the original MIT dataset and 4 modified versions of MIT by pixelization, contrast, brightness, motion blur. The MATLAB statements used to produce the above transformations are listed in Table 6. In Fig. 4 some modified images from the same sample are reported: due to the high level of stress, some images are difficult to orient even for a human.

The results in Table 5 prove the superiority of ensembles in stress conditions: while the difference of performance is very low in the

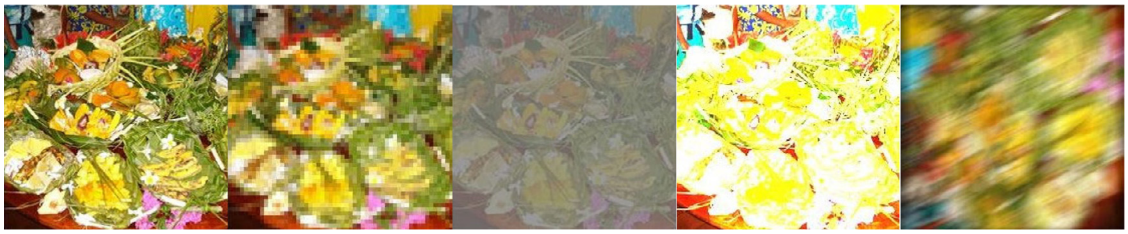

Fig. 4. From left to right: a sample image from MIT dataset, pixelation, high contrast, high brightness, motion blur.

Table 6
Image transformations applied to the MIT dataset ("im" is the input image).

Transformation	MATLAB statement
Pixelization	`imresize(imresize(im, 1/4), 4,'nearest')`
contrast	`imadjust(im,[0; 1],[0.4;0.6])`
Brightness	`im=im*3`
Motion Blur	`imfilter(im, fspecial('motion', 25, 25))`

original test set, it strongly increases in the other cases. Underspecification causes substantially different results in stress tests: while the 10 SingleStoc networks perform very similarly in MIT (as confirmed by standard deviation), they achieve different prediction performance in stress tests: in such cases it is evident the advantage of an ensemble. The variability of results is also evident comparing the two versions of MobileNet: they gain very different results under stress conditions.

Finally, we provide a comparison between our best ensembles and the state-of-the-art approaches in the literature (Table 7). The accuracies reported in Table 7 are compared with three state-of-the-art approaches (Ciocca et al., 2015; Shima et al., 2017; Swami et al., 2017) and an older handcrafted method (Lumini & Nanni, 2006) validated using the same testing protocol (some INRIA results are not present due to a different testing protocol). DenseNet is clearly the best architecture and a fusion of only 3 DenseNets (two base and one SingleStoc) performs very well. The best result is obtained fusing, by the weighed sum rule, our two best approaches based on different architectures and three DenseNet: the last part is weighted by 10 to give it more significance in the ensemble. The classification performance of this big ensemble strongly outperforms other published methods. A further increasing of accuracy can be obtained by using a reject option which discard 10% of samples which are hard to classify (as shown in the last three rows of the table).

5. Conclusions and future works

In this work we discussed a deep learning approach for the detection of image canonical orientation. Our system, based on stochastic variants of ResNet50 and other models, significantly outperforms other state-of-the-art methods in literature. As expected, it gains much better performance of traditional approaches based on hand-crafted features, but it also outperforms previous deep neural networks. Automated orientation recognition is still a hard problem, particularly in a dataset as INRIA, which includes several images with uncertain orientation, as proven by the low accuracy of all methods. However, in simpler collections including outdoor scenes the proposed method achieves an impressive accuracy higher than 98%, which is those obtained by humans.

One of the main disadvantages of deep neural networks is that the network must be very deep to reach good accuracy. To reduce the complexity of the network, we plan as a future work to investigate the influence of image resolution on the performance, to reduce the size of the input image and the number of convolution layers.

CRediT authorship contribution statement

Alessandra Lumini: Conceptualization, Software, Validation, Writing - original draft, Writing - review & editing. **Loris Nanni:** Methodology, Software, Validation, Writing - review & editing. **Luca Scattolaro:** Software, Data curation, Resources. **Gianluca Maguolo:** Conceptualization, Validation, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We gratefully acknowledge the support of NVIDIA, USA Corporation for the "NVIDIA Hardware Donation Grant" of a Titan Xp used in this research.

Table 7
Accuracy on the 4 testing datasets (comparison with literature).

Proposed	SUN	INRIA	MIT	PASCAL
Lumini and Nanni (2006)	65.73	61.34	61.04	45.40
(Ciocca et al., 2015)	81.69	–	70.52	70.64
Swami et al. (2017)	95.16	–	95.28	90.77
Shima et al. (2017)	93.70	71.83	94.55	84.10
ResNet50_FusStoc10	98.01	76.59	98.66	92.54
MobileNet_FusStoc10	97.73	76.73	98.58	91.93
DenseNet	98.08	76.86	98.36	92.67
ResNet50_FusStoch10 + Mobile_FusStoch10	98.21	76.93	98.81	93.02
DenseNet + DenseNet2 + DenseNet_SingleStoc	98.47	**78.14**	98.81	93.55
ResNet50_FusStoc10 + MobileNet_FusStoc10 + 10×DenseNet	98.47	77.46	98.96	93.55
ResNet50_FusStoc10 + MobileNet_FusStoc10 + 10×(2×DenseNet + DenseNet_SingleStoc)	**98.54**	77.60	**99.10**	**93.79**
ResNet50_FusStoc10 + Mobile FusStoc10 *with reject option 10%*	99.84	81.52	99.83	97.15
ResNet50_FusStoc10 + MobileNet_FusStoc10 + 10×DenseNet *with reject option 10%*	99.90	81.89	100	97.74
ResNet50_FusStoc10 + MobileNet_FusStoc10 + 10×(2×DenseNet + DenseNet_SingleStoc) *with reject option 10%*	99.90	82.19	99.92	97.65

Table A.1
Performance of ResNet50 varying the activation functions: accuracy on 13 medical datasets for image classification.

	Activation	CH	HE	LO	TR	RN	TB	LY	MA	LG	LA	CO	BG	LAR	Avg
	ReLu	93.54	89.88	95.60	90.00	55.00	58.45	**77.87**	90.00	93.00	85.14	94.92	88.67	87.05	84.54
	MeLU ($k = 8$)	92.92	86.40	91.80	82.91	25.50	56.29	67.47	76.25	91.00	82.48	94.82	89.67	88.79	78.94
	Leaky ReLu	89.23	87.09	92.80	84.18	34.00	57.11	70.93	79.17	93.67	82.48	95.66	90.33	87.27	80.30
	ELU	90.15	86.74	94.00	85.82	48.00	60.82	65.33	85.00	96.00	90.10	95.14	89.33	89.92	82.79
	MeLU ($k = 4$)	91.08	85.35	92.80	84.91	27.50	55.36	68.53	77.08	90.00	79.43	95.34	89.33	87.20	78.76
	PReLU	92.00	85.35	91.40	81.64	33.50	57.11	68.80	76.25	88.33	82.10	95.68	88.67	89.55	79.26
	SReLU	91.38	85.58	92.60	83.27	30.00	55.88	69.33	75.00	88.00	82.10	95.66	89.00	89.47	79.02
	APLU	92.31	87.09	93.20	80.91	25.00	54.12	67.20	76.67	93.00	82.67	95.46	90.33	88.86	78.98
MaxInput = 1	SmallGaLU	92.31	87.91	93.20	91.09	52.00	60.00	72.53	90.00	95.33	87.43	95.38	87.67	88.79	84.12
	GaLU	92.92	88.37	92.20	90.36	41.50	57.84	73.60	89.17	92.67	88.76	94.90	90.33	90.00	83.27
	softLearnable2	93.93	87.33	93.60	92.55	46.00	60.31	69.07	89.58	94.67	86.10	95.00	89.67	87.05	83.45
	softLearnable	94.15	87.44	93.40	90.36	47.00	59.18	67.73	88.33	95.00	85.52	95.52	89.33	88.26	83.17
	pdeluLayer	94.15	87.21	92.00	91.64	51.50	56.70	70.93	89.58	96.33	86.67	95.08	89.67	88.18	83.81
	learnableMishLayer	95.08	87.56	93.20	91.82	45.00	58.45	69.07	86.67	95.33	86.67	95.48	90.00	88.41	83.28
	SRSLayer	93.23	88.84	93.40	91.09	51.50	60.10	69.87	88.75	95.00	86.48	95.72	88.33	89.47	83.98
	swishLearnable	93.54	87.91	94.40	91.64	48.00	59.28	69.33	88.75	95.33	83.24	96.10	90.00	89.32	83.60
	swishLayer	94.15	88.02	94.20	90.73	48.50	59.90	70.13	89.17	92.67	86.10	95.66	87.67	87.65	83.42
	MeLU ($k = 8$)	94.46	89.30	94.20	92.18	54.00	61.86	75.73	89.17	97.00	88.57	95.60	87.67	88.71	85.26
	MeLU ($k = 4$)	92.92	90.23	95.00	91.82	57.00	59.79	78.40	87.50	97.33	85.14	95.72	89.33	88.26	85.26
MaxInput = 255	SReLU	92.31	89.42	93.00	90.73	56.50	59.69	73.33	91.67	**98.33**	**88.95**	95.52	**89.67**	87.88	85.15
	APLU	**95.08**	89.19	93.60	90.73	47.50	56.91	75.20	89.17	97.33	87.05	95.68	**89.67**	89.47	84.35
	SmallGaLU	93.54	87.79	95.60	89.82	55.00	63.09	76.00	90.42	95.00	85.33	95.08	**89.67**	**89.77**	85.08
	GaLU	92.92	87.21	92.00	91.27	47.50	60.10	74.13	87.92	96.00	86.86	95.56	89.33	87.73	83.73

Appendix

See Table A.1.

References

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2015). Learning activation functions to improve deep neural networks. In *3rd international conference on learning representations, ICLR 2015 - workshop track proceedings*.

Brahnam, S., Ghidoni, S., & Nanni, L. (2020). Ensemble of convolutional neural networks for bioimage classification. *Applied Computing and Informatics*, *17*(1), 19–35. http://dx.doi.org/10.1016/j.aci.2018.06.002.

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005). Diversity creation methods: A survey and categorisation. *Information Fusion*, http://dx.doi.org/10.1016/j.inffus.2004.04.004.

Cao, Z., Liu, X., Gu, N., Nahavandi, S., Xu, D., Zhou, C., & Tan, M. (2016). A fast orientation estimation approach of natural images. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, http://dx.doi.org/10.1109/TSMC.2015.2497253.

Chandra, A., & Yao, X. (2006). Evolving hybrid ensembles of learning machines for better generalisation. *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2005.12.014.

Cheng, Q., Li, H., Wu, Q., Ma, L., & King, N. N. (2020). Parametric deformable exponential linear units for deep neural networks. *Neural Networks*.

Ciocca, G., Cusano, C., & Schettini, R. (2015). Image orientation detection using LBP-based features and logistic regression. *Multimedia Tools and Applications*, http://dx.doi.org/10.1007/s11042-013-1766-4.

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). In *4th international conference on learning representations, ICLR 2016 - conference track proceedings*.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C., Mincu, D., & … Sculley, D. (2020). Underspecification presents challenges for credibility in modern machine learning.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, http://dx.doi.org/10.1007/s11263-009-0275-4.

Fefilatyev, S., Smarodzinava, V., Hall, L. O., & Goldgof, D. B. (2006). Horizon detection using machine learning techniques. In *2006 5th international conference on machine learning and applications (ICMLA'06)* (pp. 17–21).

Fischer, P., Dosovitskiy, A., & Brox, T. (2015). Image orientation estimation with convolutional networks. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, http://dx.doi.org/10.1007/978-3-319-24947-6_30.

Fort, S., Hu, H., & Lakshminarayanan, B. (2020). Deep ensembles: A loss landscape perspective.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Journal of Machine Learning Research*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778). http://dx.doi.org/10.1109/CVPR.2016.90.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings - 30th IEEE conference on computer vision and pattern recognition, CVPR 2017*. http://dx.doi.org/10.1109/CVPR.2017.243.

Jegou, H., Douze, M., & Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, http://dx.doi.org/10.1007/978-3-540-88682-2-24.

Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., & Yan, S. (2016). Deep learning with S-shaped rectified linear activation units. In *30th AAAI conference on artificial intelligence, AAAI 2016*.

Joshi, U., & Guerzhoy, M. (2018). Automatic photo orientation detection with convolutional neural networks. In *Proceedings - 2017 14th conference on computer and robot vision, CRV 2017*. http://dx.doi.org/10.1109/CRV.2017.59.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, http://dx.doi.org/10.1023/A:1022859003006.

Lumini, A., & Nanni, L. (2006). Detector of image orientation based on borda count. *Pattern Recognition Letters*, *27*(3), http://dx.doi.org/10.1016/j.patrec.2005.08.023.

Luo, J., Crandall, D., Singhal, A., Boutell, M., & Gray, R. T. (2003). Psychophysical study of image orientation perception. *Spatial Vision*, http://dx.doi.org/10.1163/156856803322552757.

Lyu, S. (2005). Automatic image orientation determination with natural image statistics. In *Proceedings of the 13th ACM international conference on multimedia, MM 2005*. http://dx.doi.org/10.1145/1101149.1101259.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *ICML workshop on deep learning for audio, speech and language processing*.

Maguolo, G., Nanni, L., & Ghidoni, S. (2019). Ensemble of convolutional neural networks trained with different activation functions. CoRR, abs/1905.0. http://arxiv.org/abs/1905.02473.

Misra, D. (2019). Mish: A self regularized non-monotonic neural activation function. ArXiv Preprint arXiv:1908.08681.

Morra, L., Famouri, S., Karakus, H. C., & Lamberti, F. (2019). Automatic detection of canonical image orientation by convolutional neural networks. In *2019 IEEE 23rd international symposium on consumer technologies (ISCT)* (pp. 118–123).

Nanni, L., Lumini, A., Ghidoni, S., & Maguolo, G. (2020). Stochastic selection of activation layers for convolutional neural networks. *Sensors (Switzerland)*, http://dx.doi.org/10.3390/s20061626.

Olmschenk, G., Tang, H., & Zhu, Z. (2017). Pitch and roll camera orientation from a single 2D image using convolutional neural networks. In *2017 14th conference on computer and robot vision (CRV)* (pp. 261–268).

Prince, M., Alsuhibany, S. A., & Siddiqi, N. A. (2019). A step towards the optimal estimation of image orientation. *IEEE Access*, *7*, 185750–185759.

Quattoni, A., & Torralba, A. (2009). Recognizing indoor scenes. In *2009 IEEE computer society conference on computer vision and pattern recognition workshops, CVPR workshops 2009*. http://dx.doi.org/10.1109/CVPRW.2009.5206537.

Ramachandran, P., Barret, Z., & Le, Q. V. (2018). Searching for activation functions. In *6th international conference on learning representations, ICLR 2018 - workshop track proceedings*.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*. http://dx.doi.org/10.1109/CVPR.2018.00474.

Shima, Y., Nakashima, Y., & Yasuda, M. (2017). Detecting orientation of in-plain rotated face images based on category classification by deep learning. In *IEEE region 10 annual international conference, proceedings/tencoN*. http://dx.doi.org/10.1109/TENCON.2017.8227849.

Solanki, K., Madhow, U., Manjunath, B. S., & Chandrasekaran, S. (2004). Estimating and undoing rotation for print-scan resilient data hiding. In *Proceedings - international conference on image processing, ICIP*. http://dx.doi.org/10.1109/ICIP.2004.1418684.

Swami, K., Deshpande, P. P., Khandelwal, G., & Vijayvargiya, A. (2017). Why my photos look sideways or upside down? Detecting canonical orientation of images using convolutional neural networks. In *2017 IEEE international conference on multimedia and expo workshops, ICMEW 2017*. http://dx.doi.org/10.1109/ICMEW.2017.8026216.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (pp. 1–9). http://dx.doi.org/10.1109/CVPR.2015.7298594.

Tukey, J. W. (1977). Exploratory data analysis by John W. Tukey. *Biometrics*.

Vailaya, A., Zhang, H., Yang, C., Liu, F. I., & Jain, A. K. (2002). Automatic image orientation detection. *IEEE Transactions on Image Processing*, http://dx.doi.org/10.1109/TIP.2002.801590.

Wang, Y. M., & Zhang, H. (2004). Detecting image orientation based on low-level visual content. *Computer Vision and Image UnderstandIng*, http://dx.doi.org/10.1016/j.cviu.2003.10.006.

Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., & Oliva, A. (2016). SUN database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, http://dx.doi.org/10.1007/s11263-014-0748-y.

Zhou, Y., Li, D., Huo, S., & Kung, S.-Y. (2020). Soft-root-sign activation function. http://arxiv.org/abs/2003.00547.