Overview on Max-Margin Classification: Classical and Deep Learning Applications

Anilton Cardoso Junior
UFMG
acardosoj@gmail.com

***Abstract*—The objective of this paper is to review max-margin methods in classical Machine Learning and introduce recent advances in deep learning classification with a margin maximization perspective.**

***Index Terms*—Max-margin, machine learning, deep learning**

I. Introduction

The maximal margin classifier can be defined as the hyperplane which maximizes the distance between two classes in the case of linear separability.

We can formulate a max-margin classifier with the following equation.

$$\underbrace{\max_{\mathbf{w},b} \underbrace{\frac{1}{\|\mathbf{w}\|_2} \min_{\mathbf{x}_i \in D} \left|\mathbf{w}^T\mathbf{x}_i + b\right|}_{\gamma(\mathbf{w},b)}}_{\text{maximize margin}} \text{ s.t. } \underbrace{\forall i y_i \left(\mathbf{w}^T x_i + b\right) \geq 0}_{\text{separating hyperplane}} \quad (1)$$

Support Vectors can be defined as samples where $y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) = 1$. They are the points that define the maximum margin of the decision hyperplane to the dataset. Changing the support vectors would change the hyperplane and that's not necessarily true for other samples in the dataset.

Equation 1 consists of a quadratic optimization problem. In case there's no separability, we can add slack variables to the margin constraint and minimize their norm. Equation 1 can be then formulated as in Equation 2.

$$\min_{\mathbf{w},b} \underbrace{\mathbf{w}^T\mathbf{w}}_{l_2-\text{ regularizer}} + C\sum_{i=1}^{n} \underbrace{\max\left[1 - y_i\left(\mathbf{w}^T\mathbf{x} + b\right), 0\right]}_{\text{hinge-loss}} \quad (2)$$

The second term of Equation 2 is called Hinge Loss and can be used in many other types of models besides the SVM. When we have non-linear separable data, this soft margin formulation allows some samples to be misclassified which is controlled by the parameter C.

One example of the SV solution is shown in Figure 1. In this case, the SVM found the optimal margin because the dataset is linearly separable.

Another increment in the SVM formulation is the usage of Kernel Functions. Kernels are functions that can map the data into a high-dimensional space. For example, Equation 3, which is called Radial Basis Function Kernel, maps the original input space into an infinite-dimensional one.

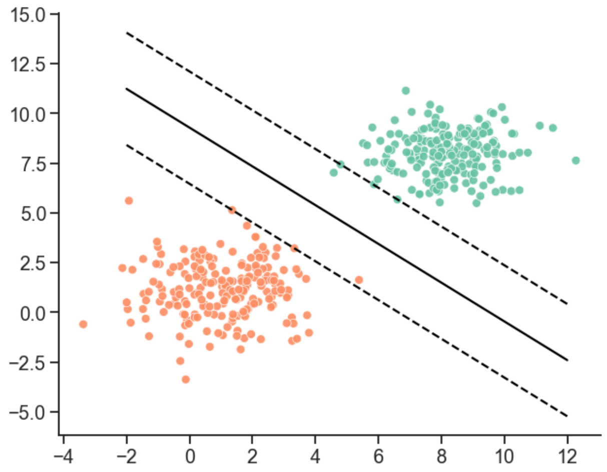
Fig. 1: two classes normal distributed and the max-margin solution

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right) \quad (3)$$

Margin maximization is important because it has a clear geometric interpretation of models which helps analyze behaviors and error analysis. It also facilitates generalization due to a more robust separation hyperplane, i.e., those classifiers are more robust to noise.

In recent advances, margin maximization has shown promising results in helping interpret deep models and enhancing robustness to imbalance and adversarial attacks.

II. Background

A. Classical Machine Learning

As described in Section I, the SVM consists of a method to find the maximum margin. Alongside the SVM there are several implementations of max-margin mechanisms in different methods, such as one of the simplest ones, the Perceptron.

We can define the following methods to maximize margin such as:

1) Regularization: As shown in Equations 1 and 2 the formulation ifself of margin maximization in SVMs consits of regularization. In fact, the margin takes an analytical form as the l_2 regularization of the parameters [4].

[12] define sufficient conditions for margin maximization in terms of loss functions and regularization.

2) Voting Strategies: As detailed in [1] and [10], voting and boosting mechanisms are very effective in terms of margin maximization. The authors explore the use of the Adaboost algorithm to analyze. Alongside with [11], they show how boosting can be viewed a kind of regularization. The authors define boosting as an incremental l_1 regularized model fitting.

[5] created a voted Perceptron to improve its margin. The method is very simple and efficient in terms of computation time. The pseudocode can be viewed in Algorithm 1. The authors also explore non-linear separable problems approaching them with kernel functions to project the problem to a high-dimensional space.

Algorithm 1 Voting Perceptron as described in [5]

```
Input: List of labeled samples $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_m, y_m)$
Output: List of weighted perceptrons $(\boldsymbol{v}_1, c_1), ..., (\boldsymbol{v}_k, c_k)$
  procedure VOTINGPERCEPTRON
      $K \leftarrow 0$
      $\boldsymbol{v}_1 \leftarrow \mathbf{0}$
      $c_1 \leftarrow 0$
      $e \leftarrow 0$
      while $e \leq$ epochs do
          for $i = 0, ..., m$ do
              $\hat{y} \leftarrow \text{sign}(\boldsymbol{v}_i \cdot \boldsymbol{x}_i)$
              if $\hat{y} = y_i$ then
                  $c_k = c_k + 1$
              else
                  $\boldsymbol{v}_{k+1} \leftarrow \boldsymbol{v}_k + y_i \boldsymbol{x}_i$
                  $c_{k+1} \leftarrow 1$
                  $k \leftarrow k + 1$
              end if
          end for
      end while
  end procedure
```

For prediction is just to take the weighted prediction for each Perceptron as in Equation 4.

$$\text{sign}\Big(\sum_{i=1}^{k} c_i \text{sign}(\boldsymbol{v}_i \cdot \boldsymbol{x})\Big) \tag{4}$$

3) Incremental Learning: [8] proposed an incremental algorithm. They define a Fixed Margin Perceptron (FMP) which finds the solution of a linearly separable problem given a fixed margin. Then they use the FMP inside an Incremental Learning framework in which they run it successively with increasing margin values.

B. Deep Learning

In recent years, Deep Learning has outperformed every known method in several domains such as image classification, natural language processing, and even in games with Deep Reinforcement Learning.

[4] reports how classical approaches to margin maximization are not appropriate for deep models, only shallow ones, and conventional methods only enforce margin maximization at the output layer. The authors show that conventional methods fail to get consistent results in generalization from small training sets, few-shot learning, and robustness to adversarial perturbations. They have proposed a margin loss function in which a large margin can be enforced in every layer. They show that a loss function with margin maximization can be viewed as:

$$\begin{aligned}&\boldsymbol{w}^* \triangleq \arg\min_{\boldsymbol{w}} \textstyle\sum_k \mathcal{A}_{i \neq y_k} \\ &\max\left\{0, \gamma + d_{f,\boldsymbol{x}_k,\{i,y_k\}} \,\text{sign}\left(f_i\left(\boldsymbol{x}_k\right) - f_{y_k}\left(\boldsymbol{x}_k\right)\right)\right\}\end{aligned} \tag{5}$$

$\mathcal{A}$ is a aggregation function, γ is the margin, $(\boldsymbol{x}_k, y_k)$ is the sample and label and d is the distance to the decision boundary. The authors argue that d is intractable for deep networks and they derive an approximation of d by linearizing it in the neighborhood of the margin. This approximation has a closed-form solution shown in Equation 6.

$$\tilde{d}_{f,\boldsymbol{x},\{i,j\}} = \frac{|f_i(\boldsymbol{x}) - f_j(\boldsymbol{x})|}{\|\nabla_{\boldsymbol{x}} f_i(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} f_j(\boldsymbol{x})\|_q} \tag{6}$$

[9] and [13] explore more benefits of encouraging large margin deep neural networks and that the widely used cross-entropy loss does not have large margin properties. More specifically the cross-entropy loss function focuses on the maximization of the model output for the true category, while the non-trivial (with non-zero margin coefficient) margin error emphasizes the gap between the model output and the true category is larger than the margin coefficient. The authors also highlight that margin-aware loss functions can lead to learning representations focused on inter-class compactness and separability between learned features.

Regarding class compactness, [6] argues that cross-entropy and softmax losses also suffer from imbalanced problems leading to poor margins. Authors propose a loss function (Equation 8) based on Gaussian Affinity metric as defined on Equation 7

$$d(\boldsymbol{f}_i, \boldsymbol{y}_j) = exp\bigg(-\frac{||\boldsymbol{f}_i - \boldsymbol{y}_j||^2}{\sigma}\bigg) \tag{7}$$

$$L = \sum_j max\big(0, \lambda + d(\boldsymbol{f}_i, \boldsymbol{y}_j) - d(\boldsymbol{f}_i, \boldsymbol{y}_i)\big) : i \neq j \tag{8}$$

[14] describes some concerning properties of deep networks such as learning input-output mappings that are quite discontinuous to a significant extent. These discontinuities make them susceptible to adversarial attacks, i.e., applying hardly perceptible perturbations to the network's input that could maximize the model's error. This makes adversarial attacks a subject of intense research in the field of Machine Learning.

[4] shows that max-margin deep neural networks can be more robust to adversarial attacks. [3] propose a training framework in which adversarial loss is minimized. The authors demonstrate a close connection between margin maximization and adversarial loss minimization. The authors propose optimizing the Objective Function defined in Equation 9.

$$\min_{\theta}\left\{\sum_{i\in\mathcal{S}_{\theta}^{+}}\max\left\{0,d_{\max}-d_{\theta}\left(x_i,y_i\right)\right\}+\beta\sum_{j\in\mathcal{S}_{\theta}^{-}}\mathcal{J}_{\theta}\left(x_j,y_j\right)\right\} \tag{9}$$

where $\mathcal{S}_{\theta}^{+}=\left\{i: L_{\theta}^{\mathrm{LM}}\left(x_i,y_i\right)<0\right\}$ is the set of correctly classified examples, $\mathcal{S}_{\theta}^{-}=\left\{i: L_{\theta}^{\mathrm{LM}}\left(x_i,y_i\right)\geq 0\right\}$ is the set of wrongly classified examples, $\mathcal{J}_{\theta}(\cdot)$ is a regular classification loss function, e.g. cross entropy loss, $d_{\theta}\left(x_i,y_i\right)$ is the margin for correctly classified samples, and β is the coefficient for balancing correct classification and margin maximization.

The authors show that their formulation results in more robustness in comparison to traditional classification and adversarial loss functions.

III. Testing Classical Approaches

A. Method

We're running different types of models in 11 datasets to measure their performances. All datasets were extracted from UCI open repository and are listed in Table I. All tests were transformed into binary classification problems by grouping different categories. These operations are shown in the Transformation column of Table I.

Dataset	Features	Instances	Transformations
Guassians	2	400	
Iris	4	150	Setosa against all
Robot LP5	90	164	Normal against all
Synthetic Control	60	600	Normal against all
Glass	9	214	Type 2 against all
Breast Cancer	30	569	
Climate Model	18	540	
Australian Credit	14	690	
Banknote	4	1372	
Ionosphere	34	351	
Parkinsons	22	195	

TABLE I: Datasets used in testing

Since we have transformed all datasets into binary classification problems, we have to deal with imbalanced issues. One of the simplest strategies to tackle this is to oversample the minority class by randomly resampling. We executed the following pipeline to all datasets before training:

- Remove features with high correlation (greater than 0.9) and high missing values ratio (greater than 0.4)
- Scale all numerical features.
- Split the data into train (80%) and test (20%) sets
- Over-sample minority class until both classes are equally distributed. We're using SMOTE, the method described in [2], instead of a simple random oversample to prevent overfitting.

We've trained SVM models with both RBF and Linear Kernels, a Voting Perceptron from [5], a simple Perceptron, a Perceptron with l_2 regularization, a Perceptron with Hinge Loss (Equation 2), a shallow MLP with one hidden layer and 10 neurons and the same MLP within an Adaboost framework.

Each model was trained 50 times on different train-test splits of all datasets and accuracy was measured for each test set.

B. Results

The results are shown in Table IV.

We also have ranked the methods using the Friedman Aligned Ranks method as described in [7]. The ranked list is shown in Table II. The score column represents the average ranking extracted from the pairwise comparisons using the Friedman non-parametric hypothesis test.

As shown, the more robust method was the Voting Perceptron from [5]. Overall, SVM using RBF kernels was robust as well.

Method	Score
Voting Perceptron	3.244
SVM RBF	3.358
MLP Adaboost	3.593
SVM Linear	4.936
MLP	5.268
MLP l_2	5.515
Perceptron Hinge	6.002
Perceptron	6.353
Perceptron l_2	6.730

TABLE II: Ranked methods

One key point to make is that there is plenty of room to improve the Voting Perceptron's implementation by using vectorized numpy functions.

IV. Deep Learning Applications

On the deep learning end, we've demonstrated the different loss functions and their results in the MNIST datasets.

We have used the same Convolution Neural Network architecture for all tests which can be seen in Figure 2. The test with Gaussian Affinity loss requires a last layer after the dense one which is a Clustering layer as described in [6].

We've systematically removed samples from classes 0 to 4 to test the robustness of these loss functions to imbalance. So, for example, MNIST 30% in Table III means we've kept 30% of samples in the minority classes. The accuracy measured was weighted one to prevent bias from imbalance.

Dataset	Softmax	Gaussian Affinity	Margin Loss
MNIST 100%	0.822 ± 0.03	0.952 ± 0.08	0.973 ± 0.02
MNIST 50%	0.801 ± 0.06	0.967 ± 0.05	0.961 ± 0.04
MNIST 30%	0.799 ± 0.11	0.960 ± 0.07	0.966 ± 0.03

TABLE III: Testing Loss Functions

The margin loss has shown better results overall and the Gaussian Affinity robustness to imbalance. As expected the softmax error has deteriorated with increasing imbalance ratios.

V. Future Work

- Test geometrical large margin methods
- Tune hyperparameters to get the best possible performance for each model
- Test robustness of deep learning loss functions to different adversarial attacks

References

[1] Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E. Schapire. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651 – 1686, 1998.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. *arXiv preprint arXiv:1812.02637*, 2018.

[4] Gamaleldin F Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. *arXiv preprint arXiv:1803.05598*, 2018.

[5] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[6] Munawar Hayat, Salman Khan, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Gaussian affinity for max-margin class imbalanced learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6469–6479, 2019.

[7] J. L. Hodges and E. L. Lehmann. *Rank Methods for Combination of Independent Experiments in Analysis of Variance*, pages 403–418. Springer US, Boston, MA, 2012.

[8] Saul C Leite and Raul Fonseca Neto. Incremental margin algorithm for large margin classifiers. *Neurocomputing*, 71(7-9):1550–1560, 2008.

[9] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, page 7, 2016.

[10] Gunnar Rätsch and Manfred K. Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6(71):2131–2152, 2005.

[11] Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *J. Mach. Learn. Res.*, 5:941–973, dec 2004.

[12] Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin maximizing loss functions. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.

[13] Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the depth of deep neural networks: A theoretical view, 2015.

[14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

Dataset	SVM RBF		SVM Linear		Perceptron		Perceptron l_2		Perceptron Hinge		Voting Perceptron		MLP		MLP Adaboost	
	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)
Gaussians	$\mathbf{1.000 \pm 0.000}$	0.043	$\mathbf{1.000 \pm 0.000}$	0.030	1.000 ± 0.002	0.033	1.000 ± 0.002	0.029	$\mathbf{1.000 \pm 0.000}$	0.033	0.999 ± 0.003	0.713	0.999 ± 0.003	8.691	0.998 ± 0.004	10.145
Iris	0.998 ± 0.007	0.026	$\mathbf{1.000 \pm 0.000}$	0.024	0.997 ± 0.013	0.028	0.992 ± 0.029	0.036	0.997 ± 0.010	0.027	0.994 ± 0.015	0.316	0.991 ± 0.020	4.429	0.998 ± 0.009	8.365
Robot	0.652 ± 0.088	0.121	0.724 ± 0.083	0.152	0.690 ± 0.085	0.062	0.685 ± 0.083	0.064	0.717 ± 0.078	0.042	$\mathbf{0.811 \pm 0.075}$	74.659	0.631 ± 0.100	4.667	0.760 ± 0.092	100.113
Synthetic Control	$\mathbf{0.993 \pm 0.007}$	0.204	0.834 ± 0.033	0.930	0.700 ± 0.082	0.091	0.710 ± 0.087	0.095	0.710 ± 0.082	0.077	$\mathbf{0.993 \pm 0.007}$	9.314	0.941 ± 0.039	12.982	0.978 ± 0.013	135.709
Glass	$\mathbf{0.728 \pm 0.079}$	0.129	0.599 ± 0.068	0.128	0.595 ± 0.094	0.067	0.565 ± 0.092	0.038	0.585 ± 0.083	0.040	0.715 ± 0.081	154.060	0.653 ± 0.092	7.181	0.726 ± 0.070	160.173
Breast Cancer	$\mathbf{0.976 \pm 0.009}$	0.214	0.970 ± 0.014	0.191	0.959 ± 0.018	0.056	0.957 ± 0.022	0.066	0.963 ± 0.018	0.072	0.974 ± 0.013	124.375	0.975 ± 0.012	12.715	0.973 ± 0.014	184.713
Climate Model	0.945 ± 0.021	0.668	0.928 ± 0.028	0.554	0.908 ± 0.031	0.049	0.885 ± 0.041	0.053	0.913 ± 0.031	0.049	0.938 ± 0.020	230.676	0.899 ± 0.035	14.452	$\mathbf{0.950 \pm 0.018}$	374.523
Australian Credit	0.854 ± 0.027	0.407	0.850 ± 0.029	0.960	0.804 ± 0.061	0.031	0.802 ± 0.056	0.033	0.821 ± 0.031	0.032	0.855 ± 0.030	602.964	$\mathbf{0.864 \pm 0.033}$	10.573	0.843 ± 0.033	249.057
Banknote	$\mathbf{1.000 \pm 0.000}$	0.308	0.984 ± 0.007	0.388	0.977 ± 0.012	0.057	0.954 ± 0.028	0.056	0.985 ± 0.008	0.069	0.999 ± 0.003	164.036	0.981 ± 0.009	19.892	0.995 ± 0.007	57.005
Ionosphere	$\mathbf{0.952 \pm 0.019}$	0.188	0.873 ± 0.035	0.289	0.839 ± 0.052	0.043	0.821 ± 0.081	0.067	0.843 ± 0.054	0.063	0.919 ± 0.025	101.543	0.880 ± 0.039	7.884	0.895 ± 0.030	178.885
Parkinsons	0.876 ± 0.046	0.061	0.862 ± 0.049	0.066	0.797 ± 0.071	0.046	0.797 ± 0.072	0.030	0.812 ± 0.055	0.050	0.870 ± 0.050	42.311	0.792 ± 0.084	4.244	$\mathbf{0.904 \pm 0.045}$	91.699

TABLE IV: Results

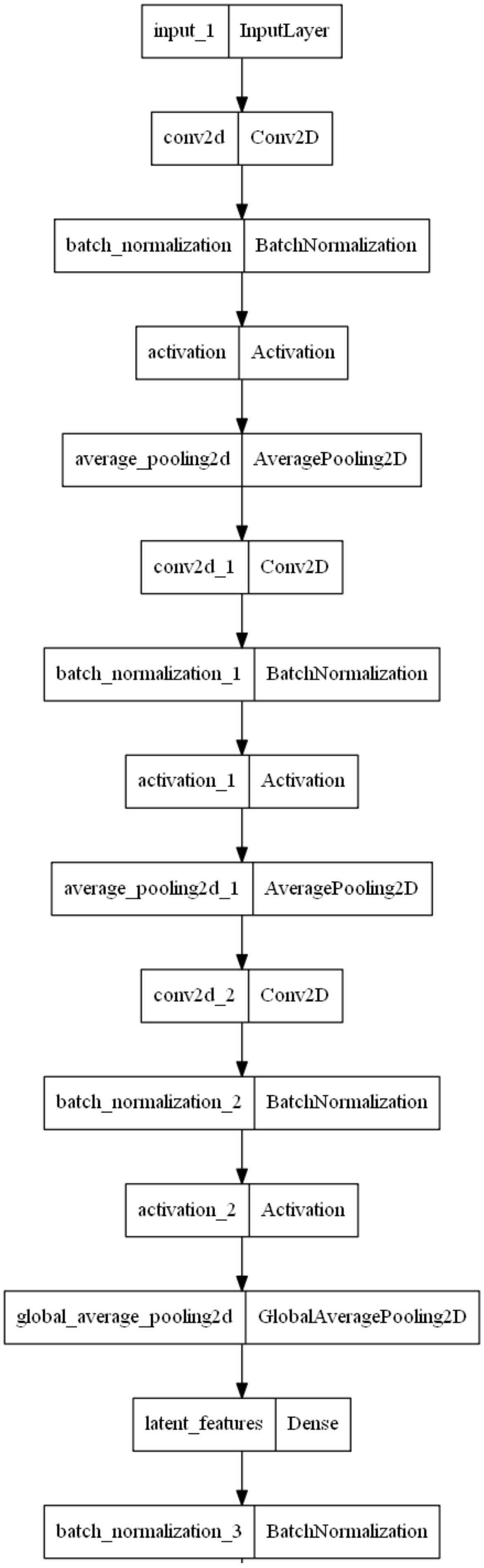
input_1 | InputLayer
conv2d | Conv2D
batch_normalization | BatchNormalization
activation | Activation
average_pooling2d | AveragePooling2D
conv2d_1 | Conv2D
batch_normalization_1 | BatchNormalization
activation_1 | Activation
average_pooling2d_1 | AveragePooling2D
conv2d_2 | Conv2D
batch_normalization_2 | BatchNormalization
activation_2 | Activation
global_average_pooling2d | GlobalAveragePooling2D
latent_features | Dense
batch_normalization_3 | BatchNormalization

Fig. 2: CNN Architecture