






A Proposal of Neural Networks with Intermediate Outputs

Billy Peralta¹(✉) , Juan Reyes², Luis Caro² , and Christian Pieringer³ 

¹ Universidad Andres Bello, Santiago, Chile
billy.peralta@unab.cl

² Catholic University of Temuco, Temuco, Chile
jreyes2011@alu.uct.cl, lcaro@uct.cl

³ INACAP, Santiago, Chile
cpieringer@inacap.cl

Abstract. The automatic data classification is an essential problem in machine learning, and it applies to different contexts such as people detection, health or astronomy. In recent years, deep neural networks have gained extensive attention due to their excellent performance on large and complex datasets. A neural network is a supervised method for classification, therefore typically requires a set of inputs and targets for the training process. However, it is possible to include auxiliary outputs that characterize aspects of the object of interest, which can accelerate the learning process. For example, in an image, a person may have extra outputs like attributes given by the presence of a hat or beard. However, the classical neural networks do not consider the presence of explicit auxiliary outputs. Furthermore, these outputs might be at a lower semantic level. We propose a framework that allows for using auxiliary outputs connected to hidden layers that complement the output connected to the output layer of the network. The key idea is to improve the training process of a neural network through a variant of the standard backpropagation algorithm that considers these auxiliary outputs. The article presents experimental evidence of the advantages of the proposed idea in various real datasets. Results also show new research venues and practical applications into image recognition considering a deep learning setting.

Keywords: Neural network · Data classification · Supervised learning

1 Introduction

Currently, artificial intelligence helps to perform most of the tedious tasks that involve classification, packaging and quality control, while human operators have

Supported by Universidad Andres Bello.

The original version of this chapter was revised: the spelling of Billy Peralta's affiliation was changed. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-31332-6_55

had to change their duties and skills [14]. Every year, researchers related to Machine Learning propose novel and powerful models able to imitate human reasoning and transfer this human capability to a machine to make it able to take decisions in unstructured environments and situations.

Machine learning (ML) algorithms are increasingly present in people's daily lives. We can usually find some applications of this type of technology in most of the giant of the IT services. For example, Google applies algorithms to improve searches ranking web pages related to the content that a user entered in the search bar. Another example is Amazon, that uses a recommender system to suggest new products based on previous purchases of a user or similar items purchased by other users [15]. Furthermore, ML also allows the development of tools based on speech recognition, such as voice assistants and chatbots [3].

One of the most essential and challenging tasks in ML is the automatic classification of data. There is a vast universe of situations at different complexity levels where this task is carried out. From problems such as written digit recognition in images [1] to identifying possible implicit risks in contracts and business agreements of large companies [9]. In recent years, Artificial Neural Networks (ANNs) have gained a new attraction to solve cases like the previous ones. ANN is one of the earlier machine learning methods to solve supervised learning problems [4, 12] simulating the brain connections [8]. The advances in computational hardware, optimization algorithms and the amount of data allow that ANN currently achieves high performances in classification tasks using deep variants [5, 11].

Traditional ANN considers a typical set of output variables associated to a set of input variables. Nonetheless, there are cases where more than one set of output variables are associated with the set of input variables; in particular, we consider the case where one set of outputs is not in the same semantic level as the typical set of outputs. For example, we can consider the people detection task where we assume the presence of a persona in a subset of training images. But, what happens if we also have access to additional visual variables, as the presence of a hat, trousers, glasses, etc., on each training image with a person?. We can consider that these visual additional variables are in lower semantic order respect to regular output variable person because these variables are typically part of a person. Typical ANN does not contemplate to handle these auxiliary set of output variables of lower semantic order. Moreover, we can think that the auxiliary output variables are hard to assign in a particular input data, however, modern classification algorithms can be reliable in many applications (for example, deep learning techniques for visual recognition), therefore, we can apply massively these techniques to assign the required auxiliary outputs. This article proposes and explores for the first time, as far we know, the idea of modeling neural networks with intermediate outputs applied to data classification task. Particularly, we research how these auxiliary output variables affect the training process. Our motivation is the possibility of training neural networks with a reduced set of data records considering more output variables instead of massive data records with typical outputs, however, the specific exploration of this idea is left to future work.

2 Standard Neural Network

An ANN generally has an input layer, a set of hidden layers and an output layer. Figure 1 shows a diagram of an ANN, where layer 1 receives the input data. The layers share information through connections between them, for example in the same figure, each node in layer 1 has a connection with all the nodes in layer 2, and the nodes in layer 2 are connected to the single node of layer 3. Each of these connections has an associated weight, and an activation function gives the response of the node.

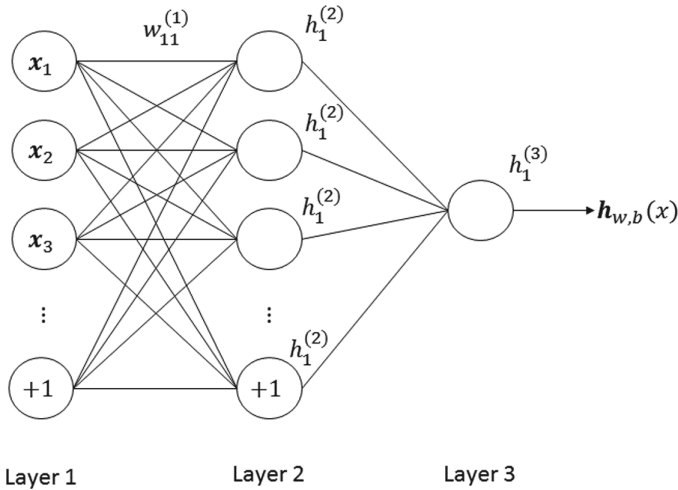


Fig. 1. Artificial Neural Network model using one hidden layer. Nodes in Layer 1 fully connect with nodes in Layer 2. Layer 3 is the output layer and concentrates all the outputs of the previous layer.

The weights associated with the links of nodes control the influence that a node in layer i has over a node in the layer $i + 1$. The $w_{ij}^{(l)}$ denotes the weight that exists between the node i at layer $l + 1$ and the node j at layer l (the current). The bias vector $b_i^{(l)}$ only has a direct connection with the next layer. Finally, the output node responds as $h_j^{(l)}$, where j denotes the node at layer l .

In an ANN, weights w and bias b are the parameters of the network. An optimization algorithm performs the searching and computation of parameters minimizing a loss function. A typical loss function is the Mean Square Error (MSE) computed between the real target (label) and the target predicted by the network. Let (x^i, y^i) be the record data of the input vector and its class used during training. The loss function for the instance i is:

$$J(w) = \frac{1}{m} \sum_{i=0}^m \frac{1}{2} \| y^i - h^{n_l}(x^i) \|^2 = \frac{1}{m} \sum_{i=0}^m J(W, x^i, y^i) \tag{1}$$

Equation 1 defines the loss function for the z -th training sample, where h^{n_i} is the output at the last layer of the network. The output indicates the predicted class y_{pred} for that sample. The optimization algorithm fits iteratively the weights such that the network increases the prediction performance. Weights $w_{ij}^{(l)}$ fit following the gradient of the loss function as define:

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha \frac{\partial}{\partial w_{ij}^{(l)}} J(w) \quad (2)$$

3 Proposed Method

Our idea consists of integrating a set of auxiliary output variables to an artificial neural network, where the main constraint is that these output variables are at a lower semantic level than regular output variables. Our rationale is that the intermediate layer can be semantically related to these auxiliary variables. For such reason, we propose to link the output of a particular intermediate layer to the auxiliary output variables considering a variation of the regular cost function of ANN by adding a new cost function of the output of a specific intermediate layer and the auxiliary output variables. In this work, we call these auxiliary output variables as *intermediate outputs*. The selection of the intermediate layer is relevant for this model; however, this work considers two intermediate layers in order to experiment with this architecture; in this case, we choose the later layer because we expect that the intermediate outputs have a nonlinear dependence of input variables. Next, we derive the solutions to the proposed modification of the typical cost function of a multilayer perceptron neural network.

3.1 Backpropagation Based on the Proposed Variant of Cost Function

In this case, we will assume that there is a h layer where the results of such neurons are compared with some auxiliary output variables. In the case of artificial vision, such outputs can be interpreted as visual attributes. This is reasonable since these attributes can be understood as intermediate patterns that allow objects to be recognized. Still more, we think that these intermediate outputs can give a good guide to the learning process of neural network. For this reason, we propose the following energy function considering the sum of the objective function and an auxiliary function weighted with λ :

$$E = \frac{1}{2}(y - y_{pred})^2 + \frac{\lambda}{2}(a - o^h)^2 \quad (3)$$

In the previous equation, we consider the mean squared error cost function by the simplicity of mathematical derivation, however, the equations are similar for case of cross-entropy loss function. In relation to the proposed neural network, we note that only the weights that reach the indicated intermediate layer consider

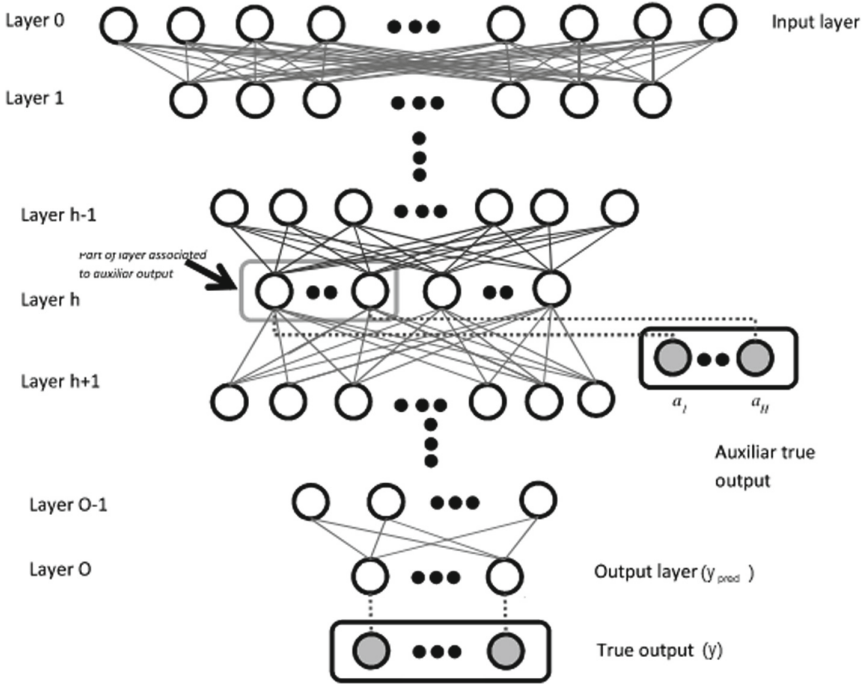


Fig. 2. Proposed variant of neural network with intermediate outputs

the error of the auxiliary output. The output layer only considers objective function error. That replicates up to layer $h + 1$. When it reaches layer h , the weights are influenced by auxiliary error (Fig. 2).

Derivating with respect to weight w_{ij} you get the same results as classical backpropagation [4] except in $\frac{\partial E}{\partial o_j}$. Again considering E as a function of output neurons and including the intermediate outputs a , we have:

$$\frac{\partial E(o_j)}{\partial o_j} = \frac{\partial E(a, net_u, net_v, \dots, net_w)}{\partial o_j} \quad (4)$$

Considering the dependence on outputs a , we have:

$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial a} + \sum_{l \in L} \left(\frac{\partial E}{\partial net_l} \frac{\partial net_l}{\partial o_j} \right) = \sum_{l \in L} \left(\frac{\partial E}{\partial o_l} \frac{\partial o_l}{\partial net_l} w_{jl} \right) \quad (5)$$

Again we assume that the subsequent derivatives are known, then we have:

$$\frac{\partial E}{\partial w_{ij}} = \delta_j o_i \quad (6)$$

with:

$$\delta_j = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} = (\lambda(o_j - a_j) + \sum_{l \in L} \delta_l w_{jl}) o_j (1 - o_j) \quad (7)$$

Therefore, we can integrate all cases in this way:

$$\frac{\partial E}{\partial w_{ij}} = \delta_j o_i \tag{8}$$

considering:

$$\begin{aligned} \delta_j &= \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \\ &= \begin{cases} (o_j - y_j)o_j(1 - o_j) & \text{if } j \text{ a a output neuron} \\ (\sum_{l \in L} \delta_l w_{jl})o_j(1 - o_j) & \text{if } j \text{ is an intern neuron} \\ (\lambda(o_j - a_j) + \sum_{l \in L} \delta_l w_{jl})o_j(1 - o_j) & \text{if } j \text{ is an intermediate output neuron.} \end{cases} \end{aligned} \tag{9}$$

As this method is based on an artificial neural network with intermediate outputs, we call it as **ANNIO**.

4 Experiments

In this Section, we test the performance of our proposed method ANNIO using diverse datasets. Specifically, we use four real data sets, two from the UCI Machine Learning Repository: Iris and Yeast [7] and two from attributes based datasets Yahoo Images (AYAHOO) and Pascal Images (APASCAL) [2]. We normalize all variables of these datasets to the range [0, 1]. All experiments are performed on a PC with 4.0 Ghz with Intel®Core™ i7-4790K CPU 8 processors with 16 GB of RAM memory. We use Python with Tensorflow to code the algorithms [10]. We compare our proposed algorithm ANNIO against the regular neural network with typical Backpropagation algorithm. We compare these techniques in terms of accuracy and convergence time. In general, we use the available test partitions to validate the results of each algorithm.

We choose the aforementioned datasets because, in addition to their corresponding input and output variables, they can have a set of variables that represent the intermediate output variables present in the analyzed objects, in another case, we generate these intermediate outputs to evaluate our proposed technique considering diverse conditions. These intermediate outputs will be entered only during the training of the proposed neural network with intermediate outputs.

In the case of Iris dataset consists of flowers data with 4 input variables and 3 classes. The intermediate output is created from the width of the sepal in centimeters of each of the flowers. These intermediate outputs are based on variable *separation*, which is the most discriminative. According to fixed intervals, we build 64 binary variables. In the Yeast dataset, we have records of yeasts with 103 inputs variables and 14 classes, which can be multiple per instance. In this case, we perform multiple experiments where in each case, we assume one original class as the output variable and the others as intermediate output variables.

In relation to image databases, the Yahoo dataset corresponds to 2037 Yahoo images related to 12 animals and also has 64 semantic binary attributes associated which correspond to our intermediate outputs, for example, if the animal has hair or tail. Finally, in Pascal dataset also corresponds to a total of 12703

images with 9751 features besides having 32 classes and where again there are 64 attributes of a semantic type which are used as intermediate outputs. Both image datasets were preprocessed by using clipped bounding box information, normalizing to 500×500 pixels and extracting the HOG features [6]. In summary, the databases are described in Table 1:

Table 1. Detail of used datasets

Database	Training set	Test set	N° of classes	N° of intermediate outputs	Number of records
Iris	90	60	3	64	150
Yeast	1500	917	2	13	2417
AYAHO	1586	1058	12	64	2644
APASCAL	6337	6347	32	64	2417

In relation to neural networks, we use full connected neural networks considering a sigmoid activation function with two hidden layers. The first hidden layer has 256 nodes and the second layer has 128 nodes. We consider a rate learning of 0.01 and 10000 training epochs. We consider this configuration for both models. In relation to ANNIO, we test by connecting the intermediate outputs to the second hidden layer, as we hypothesize a non-linear relationship between input variables and intermediate outputs.

In relation to experiments, we show the training cost for all datasets, the test accuracy and the average cost during training process. We finally evaluate the sensitivity of accuracy respect to selection of intermediate outputs and weighting parameter of intermediate outputs. In the case of the average cost of training process, it is defined as the area given by the number of epochs and cost function, in such way, we have an intuition about the learning process speed.

In general, the training process is stable for all datasets for ANNIO. For example in Fig. 3, we show the training cost evolution of ANNIO algorithm for Iris dataset, where we observe the convergence of the training process. This pattern is repeated for all datasets.

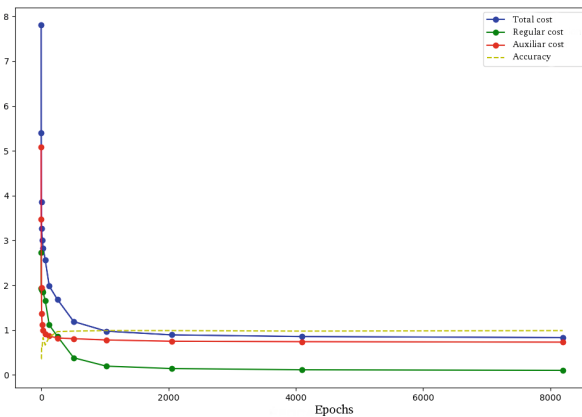


Fig. 3. Multiples costs and accuracy vs epochs for Iris dataset with ANNIO

Table 2. Results of accuracy and average training cost for all datasets

Datasets	ANN: Average cost training	ANN: Accuracy	ANNIO: Average cost training	ANNIO: Accuracy
Iris	2105	96%	1514	97%
Yeast (Class 2)	35	81%	34	93%
Yeast (All classes)	32	39%	32	46%
AYAHOO	2219	56%	1877	56%
APASCAL	5129	53%	1493	52%

We show in Table 2, we show the accuracy and the average cost of the training process obtained in all datasets considering both algorithms. We found that the reduction of the average training cost using ANNIO is significant. The results of accuracy have been variable. Particularly in Yeast dataset, ANNIO overcomes ANN by a wide 12%. In the case of image datasets, we found an acceleration of convergence but similar accuracies, however, we found that a slight tuning of weighting parameter of intermediate outputs improves ANNIO with a 2% over ANN considering APASCAL dataset as we will detail in later experiments.

We found that the use of the neural network with intermediate outputs generates a lower cost than the traditional network, obtaining a faster convergence of a correct model. Specifically, the difference of epochs for the convergence of the traditional network and the proposed network is around 128 epochs. The fact that the neural network with intermediate outputs converges early, we hypothesized that it is due to the importance of auxiliary characteristics for input data given by the weighting parameter λ , which is constrained to range $[0, 1]$.

We also explore the sensibility of accuracy respect to the number of attributes by considering only APASCAL dataset. Using Weka software [13], we rank the set of the sixty-four auxiliary attributes according to the discrimination level. Figure 4 shows the accuracy curve for both training and test partitions of APASCAL dataset. We found that the performance clearly improves in relation to the default version giving a rough improvement of 2% with respect to the total list of auxiliary attributes, which indicates that a selection of intermediate outputs can be beneficial to learning a right model. We suspect that this behaviour is originated because some attributes can produce overfitting as we can be totally uncorrelated to the classes.

Finally, we explore the cost and accuracy considering different weight parameter of intermediate outputs (λ) considering again APASCAL dataset. We use a set of auxiliary weights in the range of 0.1 to 0.9 with a step of 0.1; this set of 9 different weights was tested in ANNIO to obtain cost and accuracy curves where we showed the results in test set. The results of these experiments are shown in Table 3. We conclude that the accuracy shows an almost linear growing behaviour of the weighting parameter of intermediate outputs with respect to accuracy at least in this dataset.

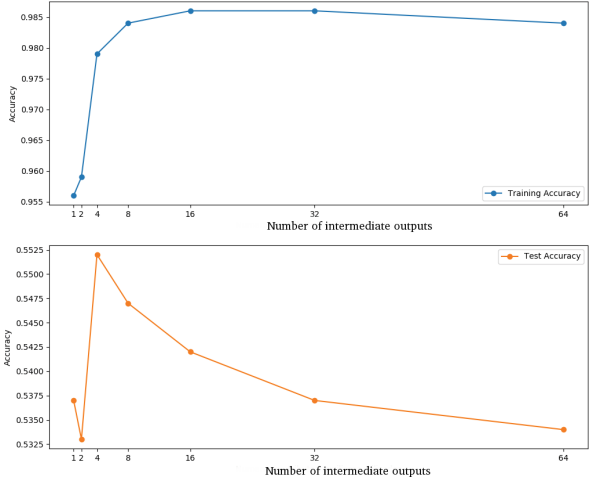


Fig. 4. Accuracy sensibility considering ordered groups of intermediate outputs.

Table 3. Sensitivity of auxiliary weights λ respect to cost and accuracy in test set of APASCAL dataset

Weighting parameter (λ)	Cost in test set	Accuracy in test set
0.1	0.644	53.1%
0.2	0.167	54.2%
0.3	0.138	54.9%
0.4	0.133	55.3%
0.5	0.128	55.8%
0.6	0.130	55.5%
0.7	0.122	56.1%
0.8	0.128	56.0%
0.9	0.137	56.4%

5 Conclusions

In this work, we propose a variant of classical multilayer perceptron neural network where we add a set of intermediate outputs which are connected to a hidden layer. The results show that network training with auxiliary variant generally obtains best accuracy results than the classical option, furthermore, it also obtains a faster convergence than the traditional neural network.

We found that not all intermediate outputs, or auxiliary attributes, have the same level of discrimination, where some intermediate outputs affect the training process. It probably is due to a lack of correlation between these auxiliary attributes and class variables. It suggests the need for using the auxiliary attributes to have a high level of discrimination.

We also found that the weight of intermediate outputs is relevant as it ponders the importance of auxiliary attributes in the adjustment of the network weights during backpropagation process. In particular, the optimal weight is the one that generates a greater improvement in data accuracy.

Respect to the future avenues of this research, we plan to test this idea considering deep convolutional neural networks and new datasets. Another point to consider is the modification of the neural network configuration, that is, modifying the number of hidden layers, number of nodes per layer and finally modifying the layer where the auxiliary outputs are connected.

References

1. Abdul, N., Amir, N.: Handwritten recognition using SVM, KNN and neural network (2016)
2. Ali, F., Ian, E., Derek, H., David, F.: Describing objects by their attributes. In: CVPR 2009, vol. 1, no. 3 p. 3.1 May 2009. <https://www.cs.cmu.edu/~afarhadi/papers/Attributes.pdf>
3. Anusuya, M., Katti, S.: Speech recognition by machine: a review (2009)
4. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press Inc., New York (1995)
5. Chollet, F.: Deep Learning with Python, 1st edn. Manning Publications Co., Greenwich (2017)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR 2005 Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 1, pp. 886–893. IEEE Computer Society, Washington (2005). <https://doi.org/10.1109/CVPR.2005.177>
7. Dua, D., Graff, C.: UCI machine learning repository (2017). <http://archive.ics.uci.edu/ml>
8. Eluyode, O., Akomolafe, D.: Comparative study of biological and artificial neural networks (2013)
9. Gao, L.: Applications of machine learning and computational linguistics in financial economics (2016)
10. Goldsborough, P.: A tour of tensorflow (2016)
11. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Cambridge (2016)
12. Gurney, K.: An Introduction to Neural Networks. CRC Press, Boca Raton (1997)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009). <https://doi.org/10.1145/1656274.1656278>
14. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
15. Smola, A.: Introduction to Machine Learning (2008)