U.P.B. Sci. Bull., Series C, Vol. 84, Iss. 3, 2022 ISSN 2286-3540

IMPROVING SMALL CONVOLUTIONAL NEURAL NETWORKS WITH SEMI-SUPERVISED LEARNING

Mihai BADEA[1], Constantin VERTAN[2], Corneliu FLOREA[3], Laura FLOREA[4], Andrei RACOVIŢEANU[5]

The widespread adoption of Convolutional Neural Networks in both academia and the commercial sector have led to a rise in interest of compact solutions in constrained scenarios. Even though CNNs with a large number of layers have shown outstanding results in various tasks, these large architectures are not always well suited in some situations. The best results can be obtained when training networks with many labeled samples, which is difficult for highly specialized tasks. Using semi-supervised learning techniques, the performance of small CNNs can be boosted to make them more practical, even when the labeled set is small. The paper focuses on three semi-supervised algorithms, used in two architectures with a small number of layers and shows how their performance can be improved when training on small datasets.

Keywords: semi-supervised learning, convolutional neural networks, image classification

1. Introduction

Although neural networks have been studied for many decades, they are still the focus of many researchers, which has led to impressive developments in their overall structure, training algorithms and handling of data. In terms of utility, in image-based applications, Convolutional Neural Networks (CNN) are currently being used in various scenarios, both academic and commercial, with continuous and notable improvements (as compared to the first popular convolutional architecture, AlexNet [1]) in the manner in which the layers behave and interact with each other.

Most of the successful applications of CNNs revolve around supervised problems, where the training algorithm uses images X_i and their corresponding

[1] Ph.D. student, University POLITEHNICA of Bucharest, Romania, e-mail: mihai_sorin.badea@upb.ro
[2] Professor, University POLITEHNICA of Bucharest, Romania, e-mail: constantin.vertan@upb.ro
[3] Professor, University POLITEHNICA of Bucharest, Romania, e-mail: corneliu.florea@upb.ro
[4] Assoc. Professor., University POLITEHNICA of Bucharest, Romania, e-mail: laura.florea@upb.ro
[5] Ph.D. student, University POLITEHNICA of Bucharest, Romania, e-mail: andrei.racoviteanu@upb.ro

labels (target values) y_i. A loss function quantifies the discrepancies between the network’s output and the target value, and an optimizer adjusts the weights which particularize the network in order to minimize the loss’ value. This method is a standard approach for many high-performance solutions, but it is not without disadvantages. The main concern when employing supervised learning is the size and adequacy of the used datasets. Typical CNN architectures proposed in the literature have millions of trainable parameters and require large collections of data to assure good generalization. For generic tasks, such as the classification of mundane objects, acquiring and labeling many samples may be straightforward, but in other tasks expert annotators are required. In other cases, gathering enough data is a challenging task in and of itself. Different approaches, such as the classical data augmentation, or such as the more recent *Mixup* method [2], can be considered solutions to the issue, generating new relevant samples used in training.

Another way to address the lack of training samples is to integrate unlabeled data alongside the labeled part. Algorithms which make use of data in this way are called **Semi-supervised Learning** (SSL) methods. Far from being a new idea, the field of SSL has seen consistent growth, partly based on the large data collections required by CNNs. Although it is not a universal solution, SSL is a useful tool to boost performance and a promising avenue to pursue.

A popular approach in improving network behavior by means of unlabeled samples is to introduce an extra regularization term in the form of a consistency loss. Methods such as the Π-model [3] and Mean Teacher [4] are noted as being effective SSL algorithms [5] which fall into this category. In other cases, the unsupervised portion of data is labeled during training and used as regular samples, Pseudo-Label [6] being a frequently mentioned example. When the extra data is used in this fashion, it acts as a form of entropy minimization [5].

A frequent point of interest in modern neural networks is the number of neural layers, denoted as the depth of the architecture. After AlexNet appeared, many notable efforts revolved around using an ever-increasing number of convolutional layers to extract highly specialized features, before the final output layers. If VGG [7] networks featured up to 19 neural layers, this number seems small when compared to ResNet [8] architectures, where values of 34, 50 or even 151 layers were possible. The increased classification accuracies observed from these networks seemed to point out towards the architecture depth as the main contributing factor for the final results. ResNet architectures even have less trainable parameters than VGG-19, further making a case for the power of depth. As a direct response to these trends, alternatives such as Wide Residual Networks (WRN) [9] show that there is a strong case to be made for a balance between depth and “width” (the number of filters in each convolutional layer). Although the number of layers is reduced, WRNs are far from being shallow and actually argue for increasing the number of parameters. Even if with GPU acceleration training large networks is not as unfeasible as it once was, one cannot overlook the fact that smaller network variants have their appeal, where applications should be more resource conscious.

This paper is focused on the analysis of different SSL solutions in the context of networks composed by very few convolutional layers. In the following sections, besides explaining the general framework of the experiments, three methods of using unsupervised data will be presented and used for two classification tasks. One of them is more generic and frequently featured when comparing Machine Learning algorithms, while the second revolves around the more specialized and difficult task of facial expression classification.

2.Architectures and proposed algorithms

All the SSL algorithms which will be used for training CNNs in the following sections rely on adding supplemental terms to the original Cross Entropy (CE) loss, typical for classification tasks. An important aspect to consider when employing such methods is that the components of the total loss need to be weighted (or balanced) adequately. If the extra term is much larger than the CE, then the training process might be affected, and classification accuracy will take a severe hit. In the other extreme, if the SSL loss is too small, then there aren’t any noticeable differences from the fully supervised baseline case.

Two architectures were used in our tests, based on the target dataset. Both architectures consist of simple blocks: a convolution layer with ReLU activation followed by max-pooling. The convolutions have kernels of size 3 x 3, with a stride of 1 and padding added as to keep the same number of rows and columns at the output. The pooling layers have 2 x 2 kernels with a stride of 2, avoiding overlapping regions within the feature maps. After these blocks, a dropout layer [10] is used before a fully connected (FC) layer, denoted as $FC_{features}$. The architectures are then ended by a typical FC output layer. Two out of the three methods make use of the features layer. Because of this, even though the networks have few convolutional layers, the dimension of that component will be initially kept at a sizeable 1024 neurons. Some variations in the architectures and the overall impact on performance will be discussed in the relevant section.

2.1. Adapted Π-model

The **first** proposal refers to a series of trainings which used a modified version of the Π-model [3] as the means of introducing unlabeled data in the process. This algorithm consists in feeding the network the same batch of images twice and enforcing a consistency loss between the outputs of the two versions (as suggested in the overall architecture shown in Fig. 1). In the original paper [3], there were two components which ensured that multiple passes of the same data would yield the same results. First, the inputs are modified before entering the network by adding Gaussian noise and other augmentations. Secondly, the use of dropout layers directly impacts the features computed throughout the forward pass. The main aim of the authors was to enforce the same network outputs, independent of the augmentations and the inherent stochastic nature of dropout layers since the original input is the same. Adding a second branch of the same architecture ensures an easy comparison of the network outputs, given the same starting images. In our implementation, the only augmentations applied on the images are random rotations of a maximum of 10° and a random horizontal flip with a probability of 50%. Also, it should be noted that the architecture proposed in [3] has considerably more convolutional layers, and the use of multiple dropout layers is expected to impact our network differently than envisioned in the original implementation. The networks which we used have either 2 or 3 convolutional layers, while the original one in [3] has 9 for all the target datasets. Furthermore, their deeper architecture should be less affected by multiple dropout layers, since the stronger features can be computed in a larger processing pipeline.

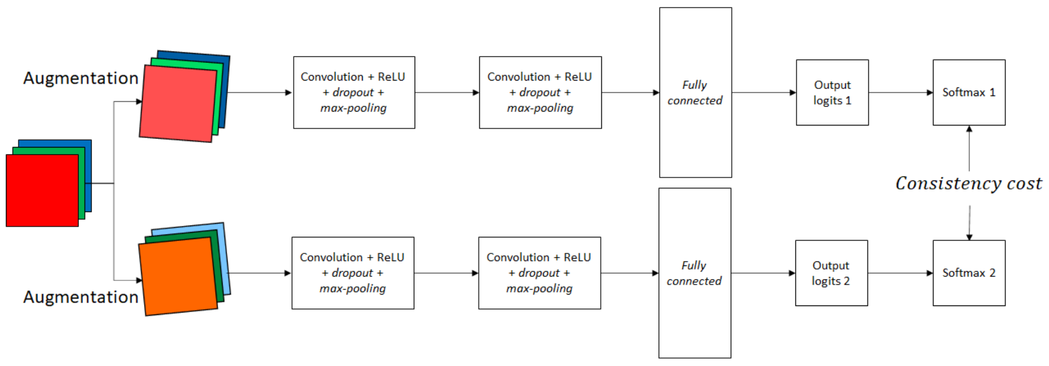
Augmentation
Convolution + ReLU + dropout + max-pooling
Convolution + ReLU + dropout + max-pooling
Fully connected
Output logits 1
Softmax 1
Consistency cost
Augmentation
Convolution + ReLU + dropout + max-pooling
Convolution + ReLU + dropout + max-pooling
Fully connected
Output logits 2
Softmax 2

Fig. 1. Implementation of the Π-model [3] on our network with two convolutional layers, where the *Consistency cost* is the second term of (1).

The error between the two versions is measured using a Mean Squared Error (MSE) loss, leading to the total loss described in (1). In it, X_i is an image with a label y_i, while o_i and o_i' are the outputs of the network after applying a softmax activation function for two versions of the input batch X_i and X_i'. In the case of unlabeled samples, only the second term of the total loss is applicable. As a supplemental experiment, a third version of the input X_i'' was tested to see if the performance can be further improved.

$$L_{total} = L_{CE}(X_i, y_i) + \alpha_1 MSE(o_i, o_i') \quad (1)$$

2.2. SSL learning via autoencoder loss

The **second** method that was tested shifted focus from the network output to the $FC_{features}$ layer. A secondary task is defined besides the regular classification one, which consists of training an autoencoder architecture which uses as input the features layer. This kind of regularization term has been used in the past, such as by Dong et al. [11], but it is not a usual semi-supervised approach. Two extra dense layers are needed, a much smaller one for compression, followed by one with the same number of neurons as $FC_{features}$ (Fig. 2). So, in the second method, the total loss is comprised of the regular CE loss and MSE loss between the input and output of the autoencoder component (2), where $FC_{features}(X_i)$ are the values of the eponymous layer given an input X_i and $o_{autoencoder}(X_i)$ is the network's output on the autoencoder branch. As was the case for the first method, when processing unsupervised batches, only the second term of the loss function is considered, since labels are not required for the reconstruction.

$$L_{total} = L_{CE}(X_i, y_i) + \alpha_2 MSE(FC_{features}(X_i), o_{autoencoder}(X_i)) \quad (2)$$

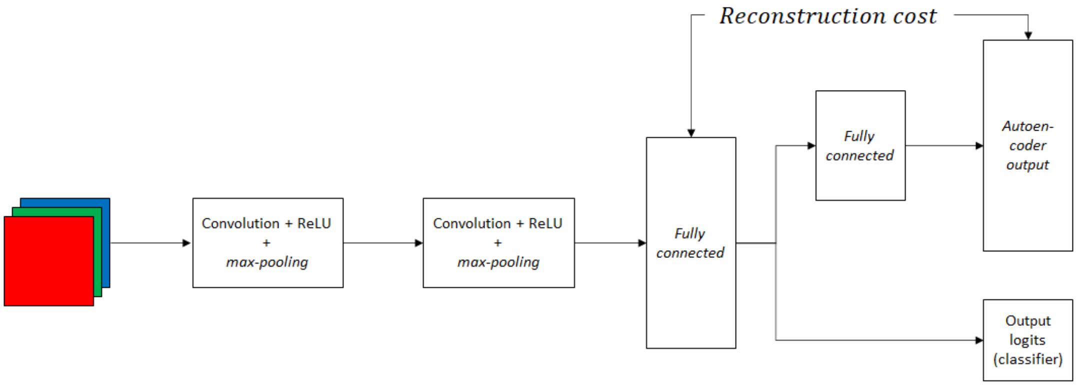
Reconstruction cost
Convolution + ReLU + max-pooling
Convolution + ReLU + max-pooling
Fully connected
Fully connected
Autoen-coder output
Output logits (classifier)

Fig. 2. Implementation of autoencoder SSL regularization on our network with two convolutional layers.

2.3. NN distance minimization for unlabeled samples

The **third** and final SSL method which was tested took inspiration from such approaches as the Center Loss [12], introduced by Wen et al., which determines features to be more discriminative by minimizing the distance to their respective class centroids. Although it was envisioned in terms of a purely supervised task, the framework can be developed to work with unlabeled samples, as it was in Florea et al. [13]. When adapted for use in SSL scenarios, a new dependency arises, forcing the introduction of labels for the unsupervised data. The method we propose disregards class prototypes and the need for extra labeling, by enforcing distance minimization to individual samples. The features of unlabeled images are compared to those of labeled ones and the Nearest

Neighbor (NN) is found. The new loss component thus minimizes the distance between the features of the two mentioned samples (as suggested in Fig. 3).

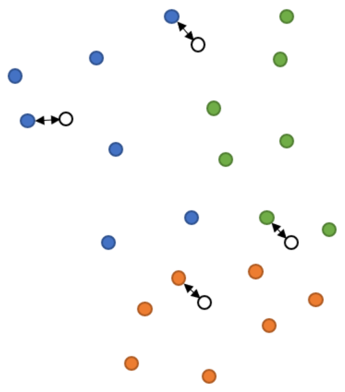

Fig. 3. The third method uses unlabeled samples
by minimizing the distance from their features to known labeled data.

Ideally, the NN should be identified from the entirety of the labeled training dataset. However, this would introduce severe overhead when the set is large, considering that distances need to be computed after each network update. Because of this, a stochastic approach was chosen, where the comparison only takes into account labeled samples from the current batch. This should not be such a troublesome issue, since most images from the same class are likely to have similar features, and the stochastic factor will have an extra regularizing effect. Compared to the previous methods, it is hard to define the total loss given a single sample, based on the nature of the supplemental term. In consequence, (3) shows the total loss for an entire batch, where $NN(X_j)$ is the Nearest Neighbor of X_j from the current batch of labeled images (X_i).

$$L_{total} = \frac{1}{N_{sup}} \sum_{i=0}^{N_{sup}-1} L_{CE}(X_i, y_i) + \alpha_3 \frac{1}{N_{unsup}} \sum_{j=0}^{N_{unsup}-1} MSE(FC_{features}(X_j), FC_{features}(NN(X_j))) \quad (3)$$

All three methods have been trained on both datasets and the same data splits to ensure a fair comparison. When necessary, different configurations and hyper-parameters will be used to ensure better results.

3. Datasets and their usage

As mentioned in the previous sections, two datasets were used in the experimental setup. The first one is CIFAR-10 [14], a popular choice when reporting classifier performance both for supervised and SSL solutions. It is comprised of 60,000 small images (32 x 32 pixels in RGB format) split between 10 classes such as *airplane*, *cat*, *frog*, or *truck*. Out of these, 50,000 are destined to be used for training and 10,000 for testing. In our scenarios, the test dataset was left in its full size, while subsets of 250, 1,000 and 4,000 training images were considered as being labeled in order to adapt the data for an SSL framework. All training images which are not part of these subsets will be automatically considered as being unlabeled.

Although test accuracy on CIFAR-10 within a fully supervised framework may go up to 99% [15], the purpose of our experiments was to study the improvement brought by the unlabeled samples in our networks with **few** layers. This approach of using a dataset usually destined for supervised training is not uncommon in SSL literature.

Since CIFAR-10 is considered a general-purpose dataset, and not especially difficult, we expanded our efforts to the more complex issue of facial expression recognition. The main extra difficulties brought by this task are related to both image composition and meaning. Since all samples must contain faces, they are bound to look more alike and the labels themselves are prone to some degree of subjectivity, based on annotator experience and background. The chosen dataset for this segment is FER+ [16], which is based on the data from FER2013 [17] but corrects various errors in the contained images and labels. The considered labels are the 6 fundamental expressions (*happiness*, *sadness*, *surprise*, *anger*, *disgust*, *fear*), along with *neutral*. In its original form, FER+ also contains *contempt*, but this expression was left out of our experiments, since it is not always included in other studies.

Each image from FER+ is a 48 x 48 grayscale image and the decision to not rescale them led to the need of the second architecture, with a supplemental convolution and max-pooling operation. This way, the number of neurons used as input for the FC segment could be kept low. From the perspective of the overall dataset size, 28,709 samples are used for training, while 3,589 are destined for validation (or *public test* as it was originally called) and just as many for the test split. In this case, subsets of 320, 400, 2,000, 4,000 and 10,000 images were considered labeled during the experiments, just as for the previous dataset.

4. Experimental results

Adding supplemental terms to the regular CE loss function can lead to the overall loss of convergence of the training process. Because of this, it is common to start training in a fully supervised fashion and add the other components after the classification loss is smaller. We tested multiple starting points considering the inherent volatility of working with few labeled samples and, indeed, starting the training with extra loss terms lead to unreliable networks. A good starting point for all experiments was anywhere above 50 epochs, possibly 100 for smaller learning rates.

Table 1

The impact of multiple dropout layers in our implementation of the Π-model. Classification accuracy [%] reported for the CIFAR-10 dataset in supervised (S) and semi-supervised (SSL) scenarios

No. images	**3 x dropout**		**FC-dropout**	
	S	**SSL**	**S**	**SSL**
250	37.1	37.63	37.7	38.76
1000	46.64	48.35	49.05	50.79
4000	59.11	58.18	62.19	62.41

For the batch of networks trained using our implementation of the Π-model, the number and position of the dropout layers proved to be an important aspect. Given the small size of networks, multiple dropout layers seemed to not improve performance. Configurations with a single, two or three dropout layers were tested. In relation to other components of the network, the regularization layer was tried after the convolution or after the pooling layer. The best results were noticed when a single dropout operation was included, before $FC_{features}$. Other variants were close to each other but with a visible drop when comparing with the wining one. In Table 1 the “3 x dropout” column signifies that three dropout layers were used, one after each convolution operation and one before $FC_{features}$. The “FC-dropout” column has a single dropout operation, before the fully connected layer. It should be reminded that the architecture in the original paper contained visibly more convolutional layers, which could explain the negative effect on performance. All three tested algorithms bring a boost in classification accuracy to various degrees. The results on the CIFAR-10 dataset are reported in Table 2. A fully supervised baseline which only uses the Cross Entropy loss is shown in the column marked “CE”.

The most interesting trends are noted for the second method, which uses the autoencoder reconstruction loss (noted as “Autoencoder Cost”), where the use of unlabeled samples negatively impacts the overall result. However, just adding the extra loss as a regularization method in a fully supervised scenario is beneficial to the network.

Table 2

The performance of the target methods on the CIFAR-10 dataset. The reported values are the classification accuracy [%] on the test set

No. images	**CE**	**Π-model**				**Autoencoder Cost**		**NN distance**	LaplaceNet [18]
		S	**SSL**	**S+3 aug.**	**SSL+3 aug.**	**S**	**SSL**	**SSL**	SSL
250	37.85	37.7	38.76	37.46	39.25	42.46	37.54	42.41	-
1000	48.73	49.05	50.79	47.83	49.55	53.19	48.24	54.1	**95.29±0.05**
4000	61.12	62.19	62.41	60.34	62.08	64.22	60.9	66.3	**95.65±0.10**

Another noteworthy aspect is that in the case of the Π-model, the extra version of the input batch (noted with “3 aug.”) does not help. When comparing the effectiveness of each algorithm, the third proposed method, noted as “NN distance”, displays the most consistent improvements with a margin of 5%, visibly better than our version of the Π-model. The last column in the table shows a comparison with LaplaceNet [18], which visibly surpasses our results, but uses a network that has 14 times as many convolutional layers and uses much stronger augmentation techniques (CutOut [19] and RandAugment [20]).

Table 3

The performance of the target methods on the FER+ dataset.
The reported values are the classification accuracy [%] on the test set

No. images	CE	Π-model		Autoencoder Cost		NN distance	MarginMix [13]
		SSL	SSL+3 aug.	S	SSL	SSL	SSL
320	50.44	51.88	51.31	52.38	52.44	52.92	50.76
400	49.16	50.38	49.7	51.49	51.34	53.58	**56.75**
2000	57.91	60.77	60.17	63.67	63.46	64.29	**60.83**
4000	66.2	65.04	64.85	67.25	68.53	70.47	**75.18**
10000	71.82	72.92	72.89	74.11	74.26	76.59	**81.25**

When analyzing the results on FER+ from Table 3, most methods display the same trends, the clear winner being, again, the third method. A notable exception is the method with the autoencoder branch, where the unsupervised set does not negatively affect the training anymore but does not bring any noticeable improvement over using the labeled data only. Our implementations are compared with MarginMix [13], featured in the last column, where the results are reported for a network more than 9 times as deep as the one we tested. Even though the difference in depth is visible, the margin is relatively small, <5% in all cases, all three methods even surpassing MarginMix when only 320 labels are available.

Table 4

The performance of the SSL method based on minimizing NN distance, with different sizes of the fully connected layer. The values represent classification accuracy [%] on FER+

No. images	FC neurons	
	1024	**128**
320	52.92	52.26
400	53.58	52.62
2000	64.29	64.14
4000	70.47	70.71
10000	76.59	77.97

Even though the approach based on NN distance minimization has the best results, the $FC_{features}$ layer still has a large number of neurons compared to the rest of the network. Since the algorithm has proved to be successful given its original constraints, further testing was done to analyze the impact of the layer's size. Reducing the number of neurons by a factor of 8 has led to interesting results, as can be seen in Table 4. Even though for smaller labeled subsets performance was slightly worse, in the case with the largest supervised part accuracy increased by >1%. This result might seem surprising at first but given the network structure most of its weights were contained by $FC_{features}$, while the three convolutional layers were not enough to produce features complex enough for the large layer. From the inference speed point of view, the network with 128 neurons in the $FC_{features}$ layer is considerably faster than other usual architectures. Using the acceleration of a smaller video card (NVIDIA GTX 1060) only 0.91ms are necessary to process a single image, while a ResNet-18 networks requires 4.39ms for the same operation and the larger network from [18] takes 8.1ms.

Of course, using CNNs with few convolutional layers is bound to reduce performance. When comparing our architecture which uses the Π-model with the one (containing 9 convolutional layers and a fully connected one) in the original paper (Table 5) there is a visible reduction in accuracy.

Table 5

The performance of the Π-model in our network compared to the original implementation. The values represent classification accuracy [%] on CIFAR-10

No. images	CE [3]	Π-model [3]	Π-model (ours)
4000	65.15 ± 1.65	87.64 ± 0.31	62.41

However, it can be seen that the margin between the fully supervised version of the network implemented in [3] and ours with the Π-model is relatively small, even though the number of neural layers is also more than twice as small, with fewer filters for the convolutional layers.

5. Conclusions

The field of CNNs has noticed many advances in recent years, but most emphasis was put on large architectures. In the experiments presented throughout this paper we have shown that there is still room for improvement in smaller networks. We have tested three different SSL solutions, and all have shown their utility in our scenarios, but the NN distance minimization method proposed as our third solution was clearly the most powerful one. Its consistent improvement of around 5% makes it a definite prospect for usage in larger, less constrained, networks as well. Semi-supervised Learning has proven to be a tool which can address many issues real-life computer vision tasks can encounter. Building on top of the established architecture may be a consistent solution, when considering the availability of existing unlabeled data, without adding any overhead at test time.

Acknowledgements

This work was partially supported by the Ministry of Innovation and Research, UEFISCDI, by the project TRANSLATE, TE 66/2020, PN-III-P1-1.1-TE-2019-0543. We gratefully acknowledge the support of NVIDIA Corporation for the donation of the Titan V GPU used for this research.

R E F E R E N C E S

[1]. *A. Krizhevsky, I. Sutskever, and G. E. Hinton.* "Imagenet classification with deep convolutional neural networks.", Advances in neural information processing systems 25, 2012, pp. 1097-1105.

[2]. *H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz*. "Mixup: Beyond Empirical Risk Minimization." International Conference on Learning Representations, 2018.

[3]. *S. Laine and T. Aila*. "Temporal ensembling for semi-supervised learning." International Conference on Learning Representations (ICLR), **Vol. 4**, No. 5, 2017, p.6.

[4]. *A. Tarvainen and H. Valpola*. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results.” In Proceedings of the 31st International Conference on Neural Information Processing Systems, December 2017, pp. 1195-1204.

[5]. *A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk and I. Goodfellow*. "Realistic evaluation of deep semi-supervised learning algorithms." Proceedings of the 32nd International Conference on Neural Information Processing Systems, **Vol. 31**, 2018, pp. 3239-3250.

[6]. *D. H. Lee*. “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks.” In Workshop on challenges in representation learning, ICML, **Vol. 3**, No. 2, June 2013, p. 896.

[7]. *K. Simonyan and A. Zisserman*. “Very deep convolutional networks for large-scale image recognition.” International Conference on Neural Information Processing Systems, May 2015.

[8]. *K. He, X. Zhang, S. Ren, and J. Sun*.“Deep residual learning for image recognition”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.

[9]. *S. Zagoruyko and N. Komodakis*. “Wide Residual Networks.” In British Machine Vision Conference, September 2016, pp. 87.1-87.12.

[10]. *N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov*. “Dropout: a simple way to prevent neural networks from overfitting”. The Journal of Machine Learning Research, **Vol 15,** No.1, 2014, pp. 1929-1958.

[11]. *W. Dong, T. Yuan, K. Yang, C. Li and S. Zhang*. “Autoencoder regularized network for driving style representation learning.” In Proceedings of the 26th International Joint Conference on Artificial Intelligence, August 2017, pp. 1603-1609.

[12]. *Y. Wen, K. Zhang, Z. Li and Y. Qiao*. “A discriminative feature learning approach for deep face recognition”. In European Conference on Computer Vision, October 2016, pp. 499-515.

[13]. *C. Florea, M. Badea, L. Florea, A. Racoviţeanu and C. Vertan.* "Margin-Mix: Semi-Supervised Learning for Face Expression Recognition." In European Conference on Computer Vision, August 2020.
[14]. *A. Krizhevsky, and G. Hinton*. "Learning multiple layers of features from tiny images", 2009
[15]. *A. Dosovitskiy, et al*. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In International Conference on Learning Representations, May 2021.
[16]. *E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang*. "Training deep networks for facial expression recognition with crowd-sourced label distribution." In Proceedings of the 18th ACM International Conference on Multimodal Interaction, October 2016, pp. 279-283.
[17]. *I. Goodfellow, et. al*. "Challenges in representation learning: A report on three machine learning contests." In International Conference on Neural Information Processing, 2013, pp. 117-124.
[18]. *P. Sellars, A. I. Aviles-Rivero, and C. B. Schönlieb*. "LaplaceNet: A Hybrid Energy-Neural Model for Deep Semi-Supervised Classification." arXiv preprint arXiv:2106.04527, 2021
[19]. *T. DeVries and G. W. Taylor.* "Improved regularization of convolutional neural networks with cutout." arXiv preprint arXiv:1708.04552, 2017
[20]. *E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le.* (2020). "Randaugment: Practical automated data augmentation with a reduced search space." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 702-703.