Artificial Intelligence Review (2021) 54:6391–6438
https://doi.org/10.1007/s10462-021-09975-1

A systematic review on overfitting control in shallow and deep neural networks

Mohammad Mahdi Bejani[1] · **Mehdi Ghatee**[1]

Published online: 3 March 2021

Abstract
Shallow neural networks process the features directly, while deep networks extract features automatically along with the training. Both models suffer from overfitting or poor generalization in many cases. Deep networks include more hyper-parameters than shallow ones that increase the overfitting probability. This paper states a systematic review of the overfit controlling methods and categorizes them into passive, active, and semi-active subsets. A passive method designs a neural network before training, while an active method adapts a neural network along with the training process. A semi-active method redesigns a neural network when the training performance is poor. This review includes the theoretical and experimental backgrounds of these methods, their strengths and weaknesses, and the emerging techniques for overfitting detection. The adaptation of model complexity to the data complexity is another point in this review. The relation between overfitting control, regularization, network compression, and network simplification is also stated. The paper ends with some concluding lessons from the literature.

Keywords Review · Neural network generalization · Overfitting · Regularization · Model simplification · Model selection · Reducing hyper-parameters · Pruning · Network compression

1 Introduction

Choosing a suitable neural network for a dataset is challenging. In the case of **underfitting**, the learning model is simple and cannot learn the data relations (Dietterich 1995), while with **overfitting**, the model is complex and only memorizes the training data with limited generalizability (Dietterich 1995; Nowlan and Hinton 1992; Hawkins 2004). In both conditions, the model cannot recognize different unseen data. For example, consider a regression problem with one independent variable x and one dependent variable y that takes values of

✉ Mehdi Ghatee
ghatee@aut.ac.ir

Mohammad Mahdi Bejani
mbejani@aut.ac.ir

[1] Department of Mathematics and Computer Science, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

Table 1. The following three polynomial models and a neural network model are stated to predict y from x:

- Model 1: $y = 0.8549x$
- Model 2: $y = 0.5x^2 - 2.2x + 7.25$
- Model 3: $y = 0.001157546x^5 + 0.000444516x^4 + 1.969512896$
- Model 4: A Multi-Layer Perceptron (MLP) with a hidden layer including 100 neurons

As shown in Fig. 1, the first model is underfitted, while the third and the fourth models are overfitted. The second model is the best fit model with small errors. The same issue occurs in classification problems.

The simplest way to solve the underfitting problem is to extend the nonlinearity of the model. However, the overfitting cannot be solved simply (Lawrence et al. 1997). Overfitting

Table 1 Data of a simple regression example with one independent variable (x) and one dependent variable (y)

Training data	x	3.54	1.80	4.87	4.92	1.46	2.94	4.30	1.34	4.67	2.65
	y	2.77	1.93	5.32	5.53	2.04	2.18	3.92	2.17	4.84	1.84
Testing data	x	1.36	1.93	2.66	3.49	3.79	1.21	3.48	4.67	2.73	4.78
	y	2.11	1.87	1.95	2.56	3.03	2.23	2.71	4.85	1.95	5.08

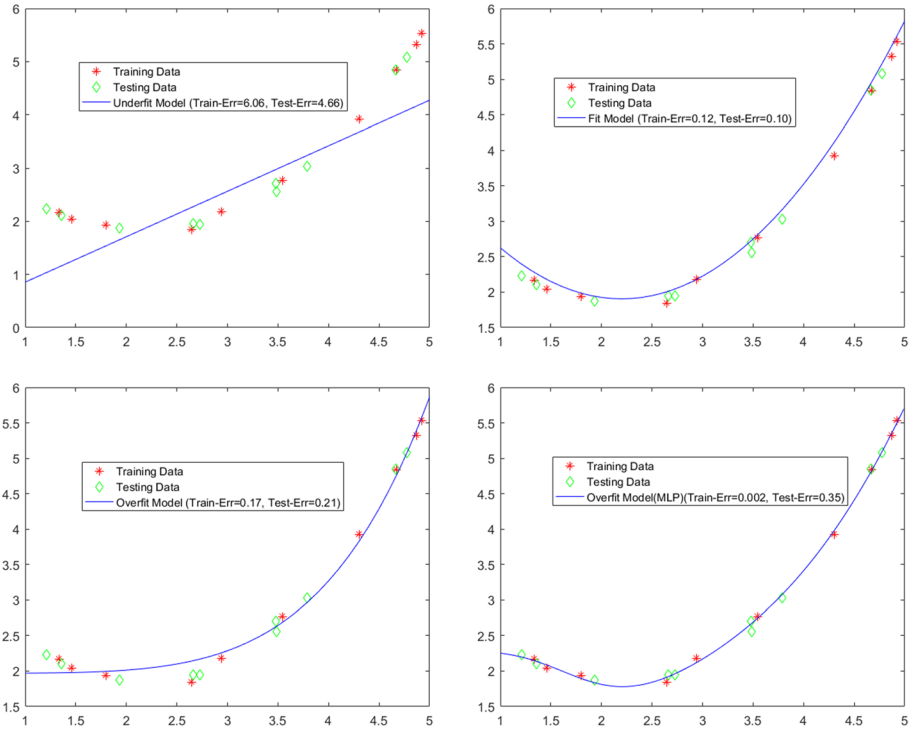

Fig. 1 Comparison between an underfitted, a fit and two overfitted regression models

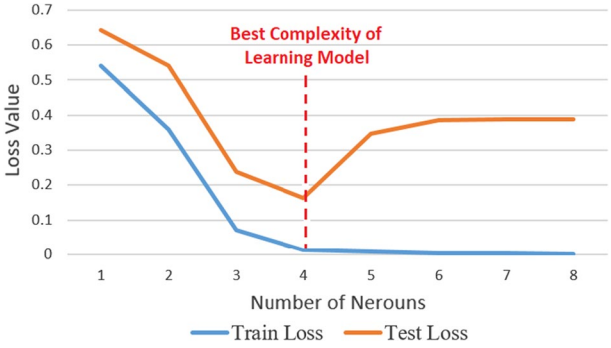

Fig. 2 Overfitting and best complexity of a learning model

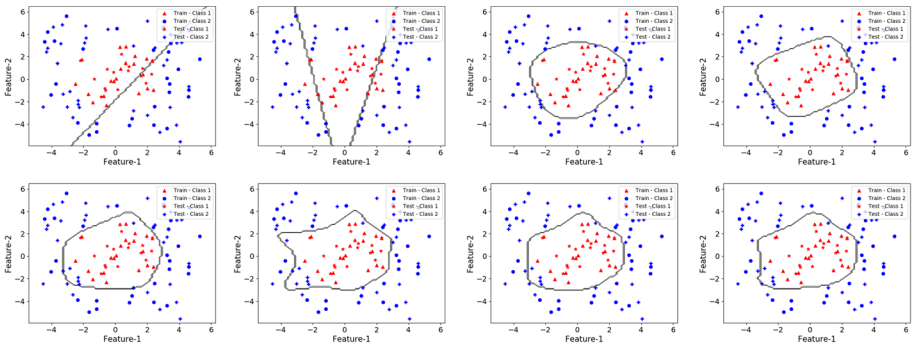

Fig. 3 Learning models for a classification problem with 2 features and 2 classes: MLP networks with a single hidden layer including 1–8 neurons (from left to right the hidden nodes are added and the networks changes from underfitting to overfitting. The fourth model is the fit)

needs model simplification by decreasing the number of free parameters, weight sharing, stopping training before overlearning, removing excess weights, decreasing redundant parameters by second-order gradient information, or penalizing the model complexity by a loss function (Nowlan and Hinton 1992; LeCun et al. 1990). For instance, low-rank factorization is a powerful method to simply learning model when the overfitting is high (Bejani and Ghatee 2020). Pattern deformation is another widely used approach in deep networks to improve the generalization power (Schmidhuber 2015).

On the other hand, overfitting is dependent on the approximation schemes (Girosi et al. 1995). For example, in Li et al. (2017) a regularized version of reinforcement learning has been used for function approximation. Classification models also approximate some separators between classes. To minimize the losses of these models on training data, one can augment the separators' parameters. Nevertheless, a great number of parameters cause overfitting (Cawley and Talbot 2007; Tzafestas et al. 1996). It is the most important reason for overfitting as clearly shown in Fig. 2. In this figure, the training and the testing losses of 8 MLP networks with a single hidden layer are presented along with 1–8 hidden neurons. Besides, the separators made by these models are illustrated in Fig. 3. Simply speaking, a model with a small number of neurons cannot learn the data, and both testing and training losses are substantial. By increasing the number of hidden neurons, the model complexity increases, while both training and testing losses diminish. When the number of neurons is

significantly large, the training loss possibly decreases, but the testing error grows because of model parameters' freedom. In Fig. 2, this happens for an MLP with 4 hidden neurons. As can be seen in Fig. 3, this network can classify the training and the testing samples of two classes with the greatest accuracy. Thus, a learning model's best complexity happens when the testing loss diverges from the training loss. This condition is useful to terminate the training process (Sarle 1995) before overfitting. Later, Sect. 3.2 defines the model complexity precisely. Then, it is possible to define a model with proper complexity for any dataset.

In the literature, the following reasons are also stated for the overfitting problem:

1. Noise of the training samples (Liu and Castagna 1999),
2. Lack of training samples (under-sampled training data) (Martín-Félez and Xiang 2014; Leung and Leung 2011; Zhao et al. 2020),
3. Biased or disproportionate training samples (Erhan et al. 2010),
4. Non-negligible variance of the estimation errors (Cawley and Talbot 2010),
5. Multiple patterns with different non-linearity levels that need different learning models (Caruana et al. 2001),
6. Biased predictions using different selections of variables (Reunanen 2003),
7. Stopping training procedure before convergence or dropping in a local minimum (Srivastava et al. 2014),
8. Different distributions for training and testing samples (Dai et al. 2007).

Any learning model should limit these overfitting sources by some overfitting controlling methods. These controllers can be categorized into three schemes, namely passive, active, and semi-active. They are defined as follows:

- **Passive schemes** search for a suitable configuration of the network before training. Sometimes, they are referred to as model selection methods or hyper-parameter optimization techniques. After designing a suitable model, its hyper-parameters remain fixed throughout the training steps.
- **Active schemes** impose a dynamic noise on the learning model or the training algorithm through the training steps, such that the model cannot memorize the details of data and the relationship between the features and outputs. These methods do not change the model architecture but activate some model components in each training step. They are also referred to as **regularization schemes**.
- **Semi-active schemes**, similar to passive schemes, change the model architecture but throughout the training steps. Sometimes, they are addressed as dynamic architecture because they reconstruct the network concerning the training statues. They follow two approaches. The first is a network construction method, which starts training with a simple network, and incrementally adds hidden units during training. The second is network pruning, which simplifies a complex network along the training process.

Figure 4 illustrates some methods which follow these schemes. Note that the boundaries between these schemes are not rigid. For example, when an active scheme imposes a dynamic noise on some network weights, it changes the model through training implicitly, and it is similar to a semi-active method. Further, in some cases, a combination of some different controllers should be assigned to solve overfitting. For example, in Heidari et al.

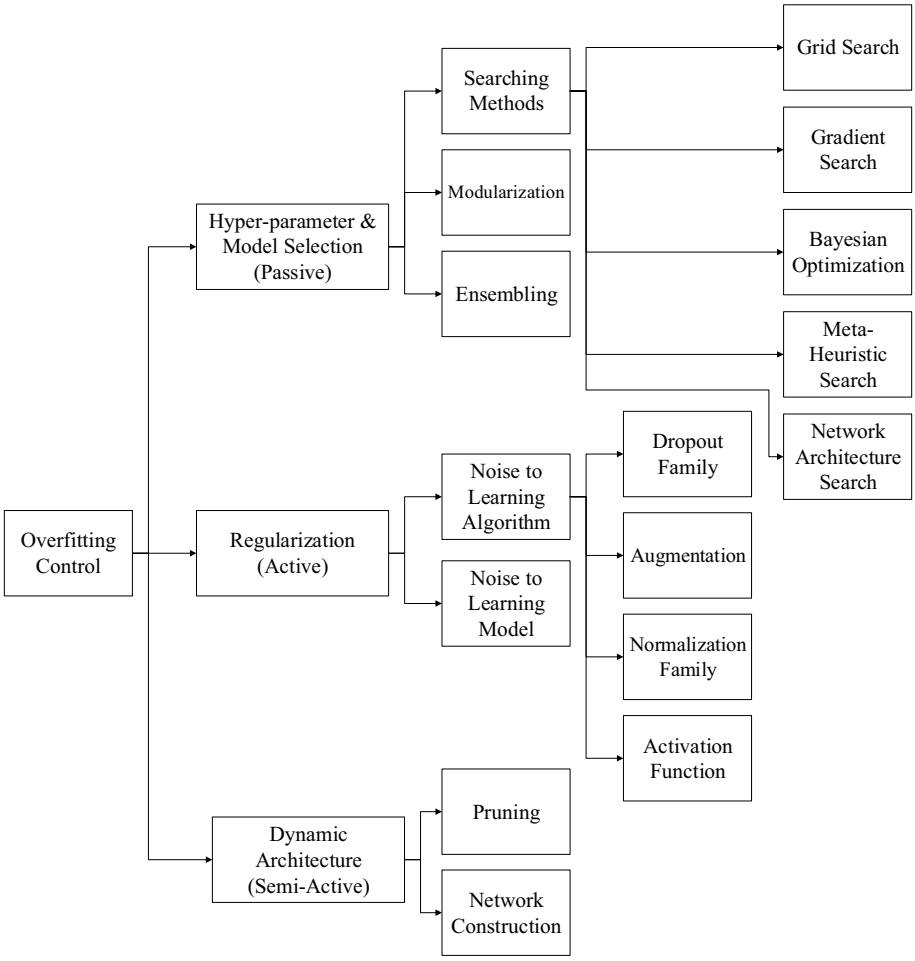

Fig. 4 The categorization of the different methods to control overfitting in neural networks

(2020) embedding-specific dropout, batch normalization, weight decay, shuffled terms are used to generalize the results. Furthermore, in some references, architecture selection is used instead of overfitting control. Architecture selection algorithms balance the complexity of a model with the required performance. Thus, the produced model fits almost all training data samples, but it cannot successfully generalize the results for unseen data. In Engelbrecht (2001), the architecture selection approaches are categorized into four groups, including brute-force, regularization, network construction, and pruning. Brute-force approaches are close to passive schemes. Regularization and active schemes are similar. The others belong to semi-active methods.

This paper reviews different controllers for overfitting and their advantages and limitations in the continuation of these works. To the best of our knowledge, there is no comprehensive review to categorize these controllers. Although there are many review papers on

shallow and deep neural networks (Ding et al. 2013; Liu et al. 2017; Li et al. 2018; Jaafra et al. 2019; Zhang et al. 2018; Darwish et al. 2020), the focus on the overfitting is limited to (Amer and Maul 2019) on modularization techniques, (Cheng et al. 2018) on model compression, (NarasingaRao et al. 2018) on some definition of overfitting, and (Bejani and Ghatee 2019) on some regularization methods. The contributions include the following:

- A comprehensive review of overfitting controlling schemes in three branches, such that one can select an efficient overfitting controller for any shallow or deep neural network.
- Summarization of the strengths and weaknesses of different methods with an in-depth discussion of their characteristics and relationships.

The rest of the paper is organized as follows. In Sect. 2, the searching methodology is presented. Section 3, presents the related topics to overfitting. Sections 4-6, include passive, active, and semi-active overfitting controlling methods. The final section ends the paper with some lessons.

2 Searching Methodology

The recommendations of the systematic review approach (Moher et al. 2009) were used to ensure the review research's reproducibility. Here, Google Scholar and Scopus were used to search the papers. The keywords were looked for in papers' titles, abstracts, and keywords. The initial keywords were chosen based on an influential paper written by Nowlan and Hinton (1992) as one of the first papers on neural network simplification and overfitting avoidance. Based on this paper, "overfitting", "overlearning", "generalization", "regularization", "simplifying neural network", "neural network simplification", "weight sharing", "model selection", "pruning of neural network", and "neural network pruning" were considered as the initial keywords. Then, the most cited papers since 2010 were taken into account, and their references were considered deeply to find the most related papers. Then, the titles, abstracts, and the keywords of the found papers were tokenized by "Keras tokenization tool" (Chollet et al. 2015). After stemming process and removing stop-words, the most frequent words were extracted, and after post-processing, the results have been presented in Fig. 5. By searching these 40 keywords, a complimentary review was performed.

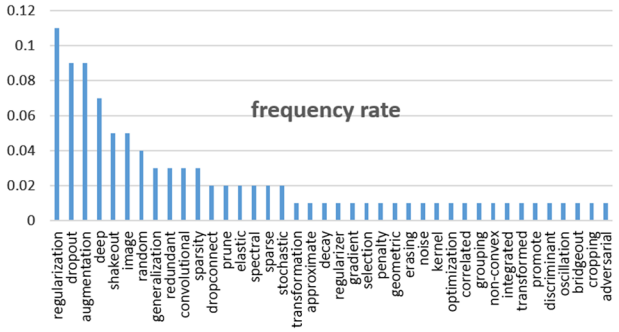

Fig. 5 The most frequent words in the titles, abstracts, and keywords of the related papers on neural network overfitting

As one can see, regularization, dropout, augmentation, deep, shakeout, image, random, generalization, redundant, and convolutional are the most 10 frequent words. These led to some important keywords related to overfitting. They also showed the importance of overfitting in deep networks, convolutional networks, and image processing problems. Finally, the words with the least frequencies were obtained. They were used as the emerging words in the relevant literature. The first 40 meaningful emerging words were as follows:

> brute-force, Bayesian, Gaussian, leverage, surpass, Dirichlet, breakthrough, decorrelation, eliminating, drop-connect, conjunction, correlations, oversized, shifted, filtering, cosine, discarding, adaptively, reflecting, compression, aggregating, lasso, elastic-net, hyper-parameter, cross-validation, robust, occlusion, flipping, re-identification, generative, meta-learning, resolution, meta-level, mini-batch, rotation, shearing, transfer, embedded, Alexnet, co-adapting

By searching on the related papers, the following emerging keywords were extracted:

> "correlation regularization", "regularization oversized neural network", "regularized negative correlation learning", "spatial-temporal regularized correlation filter", "deep neural network sparsity", "decorrelation regularization"', "gradient learning algorithm with smoothing", "gradient based adaptive regularization", "local overfitting control via leverages", and "robust deep learning"

The last search on all of the obtained keywords was done in August 2020, with no starting date. All of the obtained papers were categorized into passive, active, and semi-active schemes.

3 Related Topics to Overfitting

The overfitting happens in the gap between data complexity and model complexity. The model complexity can be stated in terms of VC-dimension (Vapnik 2006) and Bias-Variance trade-off (Geman et al. 1992). Some overviews have also been given in Hawkins (2004), NarasingaRao et al. (2018). To control the complexity of a model, there are different methods, see e.g., (Chen and Yao 2009; Li et al. 2018; Zhu et al. 2018; Liu et al. 2014; Eigenmann and Nossek 1999; Finnoff et al. 1993).

3.1 Dataset Complexity

In Ho and Basu (2002), Ho et al. (2006) for data complexity computation, three main measures have been considered, including the overlap of individual features, separability of classes, and geometry of the manifold. Their details are stated in Table 2. The last proposed method in this table simplifies the nonlinear space. These methods are known as nonlinear dimensionality reduction (Roweis and Saul 2000). The overfitting can be limited by

Table 2 Different criteria to evaluate the complexity of datasets

Ref.	Name	Idea	Complexity measure	Description
Wang et al. (2011)	Fisher's discriminant ratio	Overlap of individual feature	$\frac{(\mu_1-\mu_2)^2}{\sigma_1^2+\sigma_2^2}$	μ_1,μ_2, σ_1, & σ_2 are means and variances of two classes, respectively
Ho and Basu (2002)	Volume of overlap region		$\prod_i \frac{\min(a_{i,1},a_{i,2})-\max(b_{i,1},b_{i,2})}{\max(a_{i,1},a_{i,2})-\min(b_{i,1},b_{i,2})}$	$a_{i,j}=\max_{c_j}(f_i)$ and $b_{i,j}=\min_{c_j}(f_i)$ where f_i is ith feature, c_j is jth class
Ho and Baird (1998)	Feature efficiency		Let S_1 includes samples of the first class and S_2 contains the others. m_1 and m_2 are their means. The complexity is defined based on the intersection of the projected samples on a line connecting m_1 and m_2	
Smith (1968)	Linear separable grade one	Linear separability	$v=\min_t a^T t$ s.t. $Z^T w+t\geq b.$	v shows the grade of separability of dataset. When $a=b=\vec{1}$, the optimization problem finds the weights w to exceed the data matrix Z from b
	Linear Separable Grade Two		When outliers exist, the samples are normalized and the process is similar to Linear Separable Grade One	
Smith and Jain (1988)	Mixture of identifiability		Based on the nearest neighbors of the input samples, a minimum spanning tree (MST) is constructed. Then, the complexity is defined as the ratio of the number of neighbors that belongs to opposite classes to the number of samples Ho and Basu (2002)	
Hoekstra and Duin (1996)	Nonlinearity	Geometry of Manifold	A linear regression is defined as a separator between classes of random samples of the training data and the complexity is defined based on the error rate of this separator on the testing data, see also Ho and Basu (2002)	
Frank and Hubert (1996)	Space Covering by $\epsilon-$ Neighborhoods		For each sample x, the radius $\epsilon(x)$ gives the size of the adherence subset in the representative space. The subset of the neighbors of x is defined based on $\epsilon(x)$. After digesting smaller subsets in larger ones, it is possible to measure the level of class separability from the training samples without any tests	

removing unrelated features, and then the remaining features improve accuracy (Abpeykar and Ghatee 2019). Furthermore, in Abpeikar et al. (2020), the learning model is adapted with the data complexity.

On the other hand, some data visualization techniques enable to decrease the data complexity. As some instances, t-SNE (Maaten and Hinton 2008), LLE (Roweis and Saul 2000), Laplacian Eigenmap (Belkin and Niyogi 2002) can be addressed. They usually keep the distances between the samples. The knowledge representation in the space with a lower dimension is simpler where the unnecessary features can be neglected (Bejani and Ghatee 2018). As a wider scope, one can refer to feature extraction methods, including Principal Component Analysis (PCA), nonnegative matrix decomposition, binary decomposition, and other matrix decomposition techniques (Eldén 2019). Instead of feature extraction, a subset of features can be selected. For a review, one can refer to (Guyon and Elisseeff 2003). Feature clustering to define clusters of features with maximum relevancy and minimum redundancy is another approach to simplifying the data space. To see more details, one can refer to (Abpeykar et al. 2019).

3.2 Model Complexity

Reducing the data complexity is not applicable, but the stiffness of the model is adjustable. To compute the complexity of a learning model, VC-dimension (Vapnik and Chervonenkis 2015) states a function f of θ. On the training data $\{x_1, x_2, \ldots, x_n\}$, θ minimizes the error function of model f. When $E_{Train}(f)$ and $E_{Test}(f)$ are the error of model f on the training and the testing data, for each $\eta \in [0, 1]$, the following probabilistic inequality is met (Wittek 2014):

$$P\left(E_{Test}(f) \leq E_{Train}(f) + \sqrt{\frac{(h(\log(2n/h) + 1) - \log(\eta/4)}{n}}\right) = 1 - \eta \tag{1}$$

where h is VC-dimension of model f. The VC-dimension of a feed-forward neural network with ReLU activation function satisfies (Harvey et al. 2017):

$$h \geq WL\frac{\log(W/L)}{640}, \tag{2}$$

where W is the number of weights, and L is the number of layers. For big W and L (in deep networks), h becomes great. Thus, the probability (1) indicates that the testing error increases for any η and the generalization becomes poor. Thus, controlling both training and testing errors is needed. In reality, the testing data does not exist. It is common to apply a part of the training data to measure the generalization power as the validation data. For any training step t, the following ratio of the errors on the validation and training data can be evaluated as the overfitting level (Bejani and Ghatee 2020a):

$$v(t) = \frac{E_{Validation}(t)}{E_{Train}(t)}. \tag{3}$$

On the other hand, the complexity of neural network models can be evaluated by the condition number of the weight matrix W (Bejani and Ghatee 2020b):

$$cond_p(W) = \|W\|_p \|W^{-1}\|_p, \tag{4}$$

where $p \in \{1, 2, \ldots, \infty\}$ indicates the base for the norm. For nonlinear learning models, the condition number can be also extended (Bejani and Ghatee 2020, 2021).

4 Overfitting Control by Passive Methods

To select a model with suitable complexity, we can use model selection methods. They compare the different learning models on the validation data to choose the best model. When they use the validation data in the training process, the model probably overfits the validation data. Ng (Ng 1997) explains how overfitting occurs on validation data. He shows that if the set of hypotheses or learning models is large, folklore warns about overfitting the cross-validation data. Besides, the NAS method (Zoph and Le 2017), which chooses a controller to achieve a model with the maximum validation accuracy on the last five epochs, needs regularization methods on some small datasets. For such datasets, this reference uses a combination of embedding dropout and recurrent dropout techniques. To show what happens when the validation data is used for model selection, Table 3 presents a learning model selection example. In this example, some approximation models $p_N(x) = \sum_{n=1}^{N} c_n x^n$ estimate y for x where the training, the validation, and the testing datasets include 7, 4 and 4 samples. As the second part of this table shows, the best validation errors $E(V)$ are obtained for $N = 5$ or $N = 6$. Their training errors $E(Tr)$ are zero. Thus a model selection method chooses $p_5(x)$ or $p_6(x)$. However, as the last column shows, the testing error $E(Te)$ increases when N grows. The worst testing errors happen for $N = 5$ and 6. Thus, the best model based on the validation data cannot necessarily recognize unseen data. Meanwhile, if the testing data follow the same distribution of training data and validation data, and their sizes are great, the generalization results are reliable; see, e.g., (Nannen 2003) for more details.

In a model selector, the estimated error can be decomposed into bias and variance. A model qualifies when its bias and variance are small. To avoid overfitting in model selection, see (Cawley and Talbot 2010). Since training of deep models is time-consuming, the model selection on deep models is not very useful. For some model selection researches, see Table 4. They cover search methods, modularization, ensembling, and adaptive statistics.

4.1 Search Methods

When a learning model is in hand and can simplify without losing performance (Asadi and Abbe 2020), the simplification is useful. In other cases, a model generator produces a set of models with different architectures to learn training samples. The different searching methods evaluate the models' performances on the validation dataset, and the best one(s) are selected. The following searching methods have been widely considered in the literature. In the grid search, the different models are created based on a grid of data before the searching process, while the others define the models iteratively by a particular strategy during the searching process.

4.1.1 Grid Search

It uses a grid of parameters $\Gamma = \{\gamma_1, \ldots, \gamma_S\}$ with a discrete number of values for each hyper-parameter of the learning model. When S increases, the chance of finding better

Table 3 Approximation models $p_N(x) = \sum_{n=1}^{N} c_n x^n$ that are overfitted on the validation data

Training data										
x		−1.9319	−0.9256	−0.8534	2.0703	3.1	4.349	5.0422		
y		−0.4552	−1.3443	−1.3288	−0.4209	−0.9904	−1.2576	−0.4616		
Validation data										
x		−1.8359	−1.7602	4.246	4.8591					
y		−0.6769	−0.7897	−1.3053	−0.8149					
Testing data										
x		−2.0803	1.4067	1.7529	3.9713					
y		−0.2211	−0.3765	−0.3775	−1.3154					
N	c_6	c_5	c_4	c_3	c_2	c_1	c_0		$E(Tr)$	$E(V)$	$E(Te)$
1						0.0261	−0.9346		0.1578	0.0905	0.0707
2					0.0201	−0.0350	−1.0226		0.1508	0.0785	0.0436
3				−0.0178	0.1119	−0.0253	−1.2592		0.1302	0.0889	0.0494
4			0.0253	−0.1569	0.0503	0.6007	−0.9673		0.0002	0.0006	0.4136
5		−0.0011	0.0343	−0.1681	−0.0017	0.6639	−0.8859		0.0000	0.0005	0.4581
6	0.0001	−0.0026	0.0390	−0.1657	−0.0286	0.6757	−0.8562		0.0000	0.0005	0.4689

Table 4 Major algorithms to find optimal hyper-parameters of neural networks (Passive methods)

Ref.	Name	Description	Advantage(s) and Disadvantage(s)
LeCun et al. (2012)	Grid search	The space of the models is bounded and discretized. Thus the searching becomes so easier	The searching strategy is not optimal and is time-consuming. This method does not use the knowledge extracted from observed models
Bengio (2000)	Gradient search	Determines the training hyper-parameters of a learning model by using the gradient of the loss function	Not practical for deep networks when the second derivative of the loss function should be computed to adjust the learning rate (Larsen et al. 1996). Suitable for shallow models
Snoek et al. (2012)	Bayesian optimization	Used to find the best hyper-parameters under a prior knowledge	Using the prior distribution of hyper-parameters can increase the convergence speed
Schaffer et al. (1992), Stanley and Miikkulainen (2002)	Meta-Heuristic Search	Uses evolutionary algorithms such as genetic algorithm to find appropriate hyper-parameters of a learning model	Contains hyper-parameters and needs a new procedure to tune-up
Zoph and Le (2017)	Network Architecture Search	Searches on the different network architectures to find the best performance	The searching process is time-consuming. Besides, the founded network possibly overfitted on the validation dataset
Amer and Maul (2019)	Modularization	Includes two main goals: finding suitable sub-modules or network-blocks and an appropriate topology for these network-blocks	A hyper-problem is solved to optimize these goals. Then the learning problem is solved. Solving the hyper-problem for deep networks is time-consuming
Abbasi et al. (2016), Pashaei et al. (2019)	Ensembling	Uses a meta-learning model including multiple models that are trained (dependently or independently) and their outputs are combined to make a final decision	Designing is hard. When the models are learned separately, it is effective because each model trains once

hyper-parameters increases, and obviously, the model quality grows. In a grid search proposed by Liu et al. (2006), an evolutionary algorithm is used to search, but it can be substituted by other meta-heuristic algorithms directly. Jeng (2005) applies a competitive agglomeration clustering algorithm with a grid search to improve a regression model. Monari and Dreyfus (2002) uses a leave-one-out score for nonlinear model selection. It estimates the leverages and confidence intervals of samples during the training process and selects the best model among various models with and without equivalent complexities.

A similar approach to grid search for deep neural networks is given in Salman and Liu (2019) that trains N deep neural networks with different parameters. It also takes a predefined threshold to determine whether or not the class recognition probability is sufficient. For each sample x and model $n \in \{1, \ldots, N\}$, these probabilities for all classes are retained and their minimum is denoted with $P_n(x)$. If $max_{n=1,\ldots,N} P_n(x)$ overcomes the threshold, the corresponding model classifies x, and otherwise the model rejects x to classify.

It is worth noting that a random search is similar to the grid search that takes the samples of Γ randomly (Bergstra and Bengio 2012). When the random search takes samples by prior knowledge of hyper-parameters, it overcomes the grid search. For a random search on hyper-parameter optimization, see (Bergstra and Bengio 2012). Besides, (Bergstra and Bengio 2012) presents a model selection under uncertainty independent of the dataset. But this model causes overfitting in some cases.

4.1.2 Gradient Search

It applies a gradient descent algorithm to optimize a set of hyper-parameters of a learning model. To this end, the model selector states the learning performance in terms of model hyper-parameters implicitly or explicitly. Different linear and quadratic approximation functions are useful in this step. For example, (Bengio 2000) minimizes the following approximation loss function:

$$\min_{\theta(\gamma)} a(\gamma) + b(\gamma)^T \theta(\gamma) + \frac{1}{2}\theta^T(\gamma) H(\gamma)\theta(\gamma), \tag{5}$$

where γ shows the vector of the model's hyper-parameters, and $\theta(\gamma)$ indicates the model parameter that is a function of γ. An approximation algorithm obtains the real function $a(\gamma)$, the vector function $b(\gamma)$, and the matrix function $H(\gamma)$. However, the complexity of the Hessian matrix calculation is high. Maclaurin et al. (2015) uses a reverse-mode gradient descent instead of the Hessian matrix. Franceschi et al. (2017) also states a forward gradient-based algorithm. Larsen et al. (2012) uses a gradient method to find the weight decay method's parameters. Also, (Larsen et al. 2012) applies the gradient of the average of validation errors in k-fold cross-validation for the same purpose. Getting the gradient is not limited to supervised learning. For example, (Li et al. 2015) uses a gradient method to regularize the Least Squares Temporal Difference (LSTD) algorithm for unsupervised learning.

4.1.3 Bayesian Optimization

It finds the global solution of a learning performance function using a Gaussian Process (GP) on all available information about samples (Mockus et al. 1978). For example, assume a learning model with a hyper-parameter with different values $\theta_1, \ldots, \theta_n$. Commonly, the learning model is trained with these values separately. One can model the achieved training

errors $E_1(\theta_1), \ldots, E_n(\theta_n)$ with the following multivariate Gaussian distribution with respect to the vector of hyper-parameters $\theta = (\theta_1, \ldots, \theta_n)$:

$$E(\theta) = \begin{pmatrix} E_1(\theta_1) \\ \vdots \\ E_n(\theta_n) \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix}, \begin{pmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,n} \\ K_{2,1} & K_{2,2} & \cdots & K_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n,1} & K_{n,2} & \cdots & K_{n,n} \end{pmatrix} \right), \tag{6}$$

For $i, j = 1, \ldots, n$, μ_i indicates the mean of $E_i(\theta_i)$ on the training samples and $K_{i,j}$ is the covariance of E_i and E_j on the same samples. The Bayesian optimization techniques try to find a minimum of $E(\theta)$ by using acquisition functions to construct $E(\theta)$ from the model posterior. More practically, to simplify the computations of $K_{i,j}$, some kernel functions are presented. For more details, see (Snoek et al. 2012). Besides, (Snoek et al. 2015) applies a neural network instead of GP to accelerate the searching in the solution space. Zhao et al. (2019) proposes an adaptive and data-dependent regularization based on the empirical Bayes method. For more different applications of Bayesian techniques for hyper-parameter optimization, including evidence maximization, Bayesian model averaging, slice sampling, and empirical Bayes, see (Feurer and Hutter 2019). Furthermore, (Crammer et al. 2013) states the model parameters with the normal distribution. In the learning process, it updates the mean and the standard deviation of this distribution such that the probability of getting correct solutions increases. By sampling the network weights from the normal distribution, uncertainty increases and avoids overfitting. But this process is very time-consuming.

4.1.4 Meta-Heuristic Algorithms

Meta-heuristic algorithms such as evolutionary algorithms can be used to design a network or achieve network parameters to prevent overfitting (Ding et al. 2013; Abpeykar et al. 2019; Abpeikar et al. 2020). (Sharma et al. 2013) optimizes SVM parameters and regularizes the results with the help of the firefly algorithm, Particle Swarm Optimization (PSO), and accelerated PSO. Suganuma et al. (2017) proposes a genetic algorithm to design a Convolutional Neural Network (CNN) architecture, where the fitness is the network accuracy. Darwish et al. (2020) incorporates a dropout method in the evolutionary-deep-networks. Besides, (Fong et al. 2018) reviews on meta-heuristic algorithms for deep learning models in the context of big data analytics. This reference also points out some research directions to cover the gaps between meta-heuristics and deep learning models.

4.1.5 Network Architecture Search

Network Architecture Search (NAS) (Zoph and Le 2017) uses a recurrent neural network as a controller to find an architecture of a neural network with the highest accuracy on the validation dataset. This controller is trained by a reinforcement learning method and includes some similar blocks of some common CNN layers (convolution layers and pooling layers). This method is very time-consuming and probably overfitted on the validation data. For advanced versions of NAS, refer to (Real et al. 2019; Jaafra et al. 2019). Real et al. (2019), obtains the best architecture by using an iterative algorithm based on searching on a queue of different networks and two different mutations generating new networks.

Besides, in NAS (Zoph and Le 2017), an LSTM network is considered with two layers. In every five time-steps, the outputs of LSTM are determined to state the architecture of a layer. The LSTM network results determine the filter height, filter width, stride height,

stride width, and the number of the filters for a convolution layer. Based on the number of layers, the LSTM network generates an architecture. The designed architecture is trained on the dataset, and the performance on the validation dataset is considered as a reward signal, R, to prepare the LSTM network by reinforcement learning. In this process, the expectation of R should be maximized. Since the reward signal, R is not differentiable, (Williams 1992) uses a different policy to compute the gradient. To see a deep Q-learning, one can refer to (Mnih et al. 2013).

4.2 Modularization

A modular neural network is a neural network that breaks down into several relatively independent, replicable, and composable networks (or modules). Topological modularization acts as a kind of regularization. On the other hand, tightly coupled modules lead to overfitting. As an instance, (Kirsch et al. 2018), proposes a modularization scheme to design a network with high performance. It uses a two-step algorithm. In the first step, it defines the compositions of the modules. In the second step, it trains all modules and a controller to find better performance. Formally, this approach seeks to find the best composition of the trained modules by maximizing the following log-likelihood:

$$\mathcal{L}(\theta, \phi) = \sum_{n=1}^{N} \log P(y_n | x_n, \theta, \phi), \tag{7}$$

where θ is the parameters of the used modules, ϕ is the parameters of the controller, and $\{x_i, y_i\}_{i=1}^{N}$ is the training dataset. For a comprehensive review, see (Amer and Maul 2019).

4.3 Ensembling

By training multiple neural networks and aggregating their results, one can improve the overall performance significantly. Such ensembles of neural networks can predict unknown samples more accurately than the individuals (Li et al. 2018; Bejani and Ghatee 2018; Abbasi et al. 2016). In Abbasi et al. (2016) an ensemble of Radial Bases Function (RBF) networks is stated. In Abpeykar and Ghatee (2019), Abpeykar et al. (2019), different ensembles of neural networks in the decision tree structures are investigated. In Abbasi et al. (2016) a regularization term is added to the ensemble method. In Abpeykar et al. (2019), Abpeykar and Ghatee (2019) clustering algorithms are used to decompose the features into some subsets to reduce redundancy and increase dependency. This approach simplifies data space and controls overfitting in high-dimensional datasets. Also, in Abpeykar and Ghatee (2019), a cut point is used in a neural tree to decrease the computation time and prevent overfitting. Besides, in Abpeikar et al. (2020), an expert system is developed to control overfitting. This expert system that is located in any node of the neural tree evaluates data complexity and selects a suitable neural network. A meta-heuristic algorithm is also used to define a reasonable set of features to avoid overfitting when it is high.

4.4 Conclusions and Future Works on the Passive Schemes

The passive methods are some meta-models that create learning models. After training, it evaluates the performance of the learning models on the validation data. Based on these evaluations, it updates the learning models. One can repeat this procedure until achieving

a suitable model for a learning task. Passive methods' problems are their long computation time and overfitting on the validation data. For solving the first problem, someone uses prior knowledge to initialize the model parameters or modify the searching process (Yam and Chow 2000; Ivanova and Kubat 1995). For solving the second, it is possible to use random sampling in the training process (Bergstra and Bengio 2012). The following topics are important when choosing a passive method:

- They add some hyper-parameters to the learning models, and finding hyper-parameters is hard.
- These methods can be used on shallow models usually, as they contain few parameters.
- These methods are very time-consuming for applying on deep models.
- They produce models possibly overfitted to the validation data.
- By combining randomness in the passive methods, the results probably improve, although the processing time remains long.

Using the uncertainty and prior knowledge in the process of model selection can be considered in future works. To accelerate the processing time of passive schemes, one can decrease the number of hyper-parameters and use the previous results to limit the searching space. Considering multi-criteria decision-making techniques for model selection can also be followed, but it seems useful for shallow models. To date, no effective model selection method has been proposed for deep learning models.

5 Overfitting Control by Active Methods

Regularization methods simplify complex models along the training process. They impose dynamic noises on the model parameters or the training parameters (Blanc et al. 2020). It causes control overfitting. Figure 6 displays the published papers' trend on the regularization methods. Because of the importance of these methods, in what follows, the related references are compared substantially.

5.1 Imposing Noise to the Learning Model

Tikhonov and Arsenin (1977) have proposed this method to find a stable solution for ill-posed systems. In Natterer (1984) the error bound of Tikhonov regularization has been

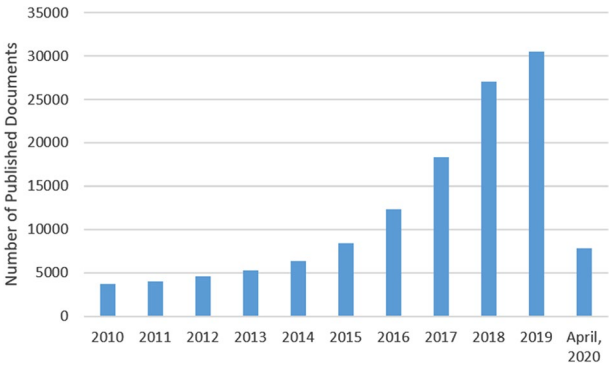

Fig. 6 The number of published papers on "regularization" + "neural network" based on the Google-Scholar database

determined. In Scherzer et al. (1993) a posterior strategy has been presented for choosing the regularization parameters of the Tikhonov model. Some applications have been also given in Golub et al. (1999), Beck and Ben-Tal (2006), Calvetti and Reichel (2003), Nashed (1986). Because the loss function of a neural network can be minimized by solving a nonlinear system, the Tikhonov method can be extended to construct a fit neural network. Any stable solution of a nonlinear system leads to a learning model with low complexity. Based on (Bishop 1995), Tikhonov regularization for neural network training is equivalent to the training with noise augmentation. Generalized Tikhonov methods are also considered in Vauhkonen et al. (1998). In Bejani and Ghatee (2020b), the effect of Tikhonov term in CNN training is analyzed, and it is combined with Singular Value Decomposition (SVD) of weights. In Bejani and Ghatee (2020), Tikhonov term is incorporated with low-rank matrix decomposition. To present some details, consider the following empirical error for learning model f on the training samples $\{x_i, y_i\}_{i=1}^{D}$:

$$\min_{f} E(f;\{x_i, y_i\}_{i=1}^{D}). \tag{8}$$

To train these D samples, the empirical error should be minimized to achieve the hyperparameters (weights) of f. Meanwhile, the error of a model on unseen data is important to judge the quality of the learning model. As Fig. 1 shows, sometimes the complexity of the model is not proportional to the data complexity. To adapt these complexities, one can add a regularization term to the empirical error as the following:

$$\min_{f} E^{*}(f;\{x_i, y_i\}_{i=1}^{D}) = E(f;\{x_i, y_i\}_{i=1}^{D}) + \lambda R(f), \tag{9}$$

where, the nonnegative function $R(f)$ measures the complexity of the learning model f. The closer f to a linear function, the more $R(f)$ converges to 0. Conversely, for a non-linear learning model f, the value of $R(f)$ is high. This function can be stated as the following:

$$R(f) = \int \left| \frac{\partial^m f}{\partial^m x} \right| dx \tag{10}$$

By setting $m = 1\,(m = 2)$ in Eq. 10, $R(f)$ interprets the summation of values of the first (the second) derivative of the learner model f. The coefficient λ adapts the regularization effect and depends on the complexity of data. In many cases, data complexity cannot be defined precisely. Instead, $\lambda R(f)$ changes randomly to add a noisy effect on the loss function of the learning model. The optimal solution of Problem 9, leads to a simple learning model with the minimum empirical error. This function does not memorize the training data, and hence the overfitting diminishes. In Table 5, some researches on Tikhonov regularization that impose the noise on the learning model have been described. This table covers three main approaches. The first approach leads to the minimization of the network weights magnitudes. The second increases the correlation between the network weights. The third reframes the inputs with respect to the outputs.

Furthermore, imposing the noise to the learning model can also be defined independently of the Tikhonov term. Optimally Pruned Extreme Learning Machine (OP-ELM) (Miche et al. 2011) with Gaussian kernel is an example. In Belkin et al. (2006) a family of learning algorithms based on the manifold regularization has been developed by exploiting the geometry of the marginal distribution. Also, in Abbasi et al. (2016) a weight decay regularization term for an ensemble learning model is augmented.

Table 5 Active methods with imposing noise to learning models (The Tikhonov regularization schemes)

Refs.	Schemes name	Tikhonov term	Description
Krogh and Hertz (1991)	Weight Decay	$\|\|w\|\|_2^2$	By minimizing $\|\|\vec{w}\|\|_2^2$, the input weights of some neurons become close to zero and they will be neglected
Bejani and Ghatee (2020a)	Adaptive Weight Decay	$\alpha(t)\|\|w\|\|_2^2$	Similar to weight decay but its coefficient $\alpha(t)$ adapts with training process by using some expert rules
Bejani and Ghatee (2020b)	Adaptive Spectral Regularization(ASR)	$\|w - w^*\|_2^2$	In each training step, w^* shows the SVD approximation of the best-found weights on the validation data, and the Tikhonov term converges w to w^*
Tomar and Rose (2014)	Manifold	$\sum_{i,j} \|\|y_i - y_j\|\|(w_{i,j}^{int} - w_{i,j}^{pen})$ where $w_{i,j}^{int} = e^{(-\frac{\|\|x_i - x_j\|\|_2^2}{\rho})}$ iff $e(x_i, x_j) = 1$ & $C(x_i) = C(x_j)$. Otherwise, it is 0 $w_{i,j}^{pen} = e^{(-\frac{\|\|x_i - x_j\|\|_2^2}{\rho})}$ iff $e(x_i, x_j) = 1$ & $C(x_i) \neq C(x_j)$ and otherwize it is 0. $C(x)$ shows a cluster including x	The idea is inspired from Laplacian Eigenmaps Belkin and Niyogi (2003) and uses k-NN algorithm to define $e(x_i, x_j) = 1$ iff x_i and x_j are neighbours. $min \sum_{i,j} \|\|y_i - y_j\|\|(w_{i,j}^{int} - w_{i,j}^{pen})$ brings outputs closer when the inputs are close
Ayinde et al. (2019)	Enhancing Diversity in Feature Extraction	$\sum_{l=1}^{L} \sum_{i=1}^{n_l} \sum_{j=1}^{n_l} (\Omega_{i,j}^{(l)}(w))^2 M_{i,j}$, where $\Omega_{i,j}^{(l)}(w)$ is the correlation between weights i and j in layer l and $M_{i,j}$ is a fixed mask. n_l is the number of nodes in layer l	By minimizing this term, the correlation between the layers weights increases, and the layers can extract diverse features. It is very useful in convolution layers
Ma et al. (2019)	Transformed ℓ_1	$\sum_l \mu_l T_{\ell_1}(w^{(l)}) + \bar{\mu}_l \sum_g \|w_g^{(l)}\|_2$, where $\mu_l > 0$, $\bar{\mu}_l = 1 - \mu_l$, $T_{\ell_1}(w) = \sum_i^g \frac{(a+1)\|w_i\|}{a+\|w_i\|}$, and $a > 0$	By minimizing this convex regularization term for every layer l, some of the unnecessary weights are removing
Zou and Hastie (2005)	Elastic Net	$\alpha\|w\|_1 + \beta\|w\|_2$, where $\alpha, \beta \geq 0$	Similar to weight decay
Phan and Le Thi (2019)	Difference of convex functions	$\sum_{j=1}^{J} \eta_\alpha(\|w^j\|_p)$ where $\eta_\alpha(t) = min\{1, \alpha\|t\|\}$ and α is a tuning parameter such that $\eta_\alpha(t)$ approximates the step function as α tends to infinity	In each iteration, the concave part of the regularization function is approximated by its affine majorization and by minimizing the produced convex functions, the sparse discriminant vectors interpret data well

Table 5 (continued)

Refs.	Schemes name	Tikhonov term	Description
Wu et al. (2014)	Smooth ℓ_2	$f(\|\|w\|\|_1)^{0.5}$, where $f(x) = \begin{cases} \|x\|, & \text{if } \|x\| \geq a \text{ and} \\ -\frac{1}{8a^3}x^4 + \frac{3}{4a}x^2 + \frac{3}{8}a, & \text{if } \|x\| < a \end{cases} a > 0$	By minimizing $f(\|\|w\|\|_1)$ as a piecewise polynomial approximation of $\|w\|_1$, the synaptic weight leads to zero smoothly
Tam and Dunson (2020)	Fiedler	$F(\|W\|)$ that is the Fiedler value of the neural network's underlying graph and $\|W\|$ is the absolute of the weight matrix	It is equivalent to a structurally weighted ℓ_1 penalty and causes sparsity induction
Nabian and Meidani (2020)	Physics-Driven	$\frac{1}{2n}\sum_{i=1}^{n} N(x_i, \tilde{u}(x_i, \theta))^2$, where $N(x_i, \tilde{u}(x_i, \theta))$ measures the divergence of the network solution $\tilde{u}(x_i, \theta)$ from the governing laws at input location x_i	The learning task considers both the mean squared error and the divergence from the governing laws

5.2 Imposing Noise to the Learning Algorithm

These schemes implicitly affect the models and the training dataset. They can be referred to as non-Tikhonov regularization and are categorized as follows.

5.2.1 Dropout Family

The dropout (Srivastava et al. 2014) is the most popular scheme to control overfitting in the training processes. This method considers a Bernoulli probability p in each training iteration to decide whether or not the training process updates neurons' weights. The dropout scheme can be represented as a Tikhonov method by using the following complexity function (Demyanov 2015, P. 66):

$$R(f) = \frac{1-p}{p}||\Lambda w||_F^2 \tag{11}$$

where p is the keep probability, Λ is a diagonal matrix of eigenvalues of the covariance matrix of the dataset, and w is the weight vector of the linear model. In recent works, Low-Rank matrix Factorization (LRF) has been used to drop out some parameters of a learning model along the training process when the complexity of each layer is high (Bejani and Ghatee 2020). This method, which is entitled "Adaptive LRF", simplifies the network only when it is needed. In Bejani and Ghatee (2021), Laliga is defined that applies the least auxiliary loss-functions together adaptive weights to regularize VGG neural networks.

In another approach, the noise is imposed on inputs. The constant noise leads to overfitting because, in the training process, the model learns this noise as a principal part of the input data. By imposing the random noise on the input data in each iteration of the learning process, the model cannot learn the constant noise of the input data. As an instance, the whiteout scheme (Li and Liu 2016) adds the following noise to the inputs of layers:

$$\tilde{x}_i^{(l)} = x_i^{(l)} + e_i \tag{12}$$

where the components of $e_i = (e_{i,j})$ are stated with the following normal distribution:

$$e_{i,j} \sim N(0, \frac{\sigma^2}{|w_{i,j}^{(l+1)}|^\gamma} + \lambda) \tag{13}$$

$\gamma \in [0, 2]$, $\sigma > 0$, and $\lambda > 0$ are the parameters of this method. This regularization method changes the inputs in each epoch and does not allow the network to memorize the training data. For other extensions, see (Wan et al. 2013; Kang et al. 2018; Khan et al. 2018; Krueger et al. 2017; Larsson et al. 2017; Khan et al. 2019; Liu et al. 2020). In Zhao et al. (2020) wavelet transforms of data are incorporated into the training process to produce a set of diverse data when the training data are limited. Discrete wavelet transforms are also used by Eftekhari and Ghatee (2018) to extract the features and to improve generalization. In Heidari et al. (2020), the SVD of weights are used to produce diverse patterns to improve generalization.

In another approach, reweighting algorithms are used on the samples to control overfitting. In Ren et al. (2018), a meta-learning gradient-based algorithm is defined to

assign weights to training samples of a mini-batch to minimize the loss on a clean, unbiased validation dataset.

5.2.2 Augmentation Family

The augmentation of a dataset is a popular method to increase the number of samples and to increase generalization power. In some instances (Kwasigroch et al. 2017; Wąsowicz et al. 2017; Galdran et al. 2017), the experimental results of the affine transition augmentation have been presented. Also, in Jin et al. (2015), the effect of the noise on the input of models has been discussed. In Yang et al. (2020), the gradient augmentation is defined that works on randomly transformed training samples to regularize a set of sub-networks to learn well-generalized and more diverse representations. Besides, the overfitting in graph-structured convolutional neural networks has been presented by Kipf and Welling (2017). In Verma et al. (2019), a regularized training scheme for graph neural network is developed by using data augmentation. In Table 6, some important augmentation schemes have been expressed.

On the other hand, a Generative Adversarial Network (GAN) (Goodfellow et al. 2014) enables raising the number of samples. GAN models include convolutional based GAN (Yu et al. 2017), condition-based GAN (Mirza and Osindero 2014), and autoencoder based GAN (Donahue et al. 2017). See (Pan et al. 2019) for more details. The application of GAN for data augmentation can be seen in Antoniou et al. (2017). Also, in Perez and Wang (2017), a GAN has been trained to generate different styles in a dataset.

5.2.3 Normalization Family

A simple but effective regularization scheme is batch-normalization (Ioffe and Szegedy 2015). This scheme has been proposed for solving the internal covariate shift, which occurs in neural networks. Each batch of training samples produces a new distribution in the output of each layer of the network, and each layer has to learn the new distribution per batch data. In Luo et al. (2019), it is shown that this scheme can control overfitting problems. More researches have been stated in Arpit et al. (2016), Ioffe (2017), Wu and He (2018), Ba et al. (2016), Heidari et al. (2020). In a more different approach, the weights of

Table 6 The augmentation schemes for increasing the samples and decreasing overfitting

Refs.	Description
Patel et al. (2019)	Finding the cuts of shapes to generate new data
Mikołajczyk and Grochowski (2018)	Transferring the image styles
Salamon and Bello (2017)	Combining four sub-methods, including 'Time Scratching','Pitch Shifting', 'Dynamic Range Compression', and 'Background Noise'
Taylor and Nitschke (2018)	Using geometric and photometric techniques to transform the shapes and shifting their pixels
Zhong et al. (2020)	Exchanging a random region of the input with some noise
Cubuk et al. (2019)	Using the reinforcement learning to learn the augmentation strategy
Verma et al. (2019)	Interpolating hidden states by generating new images based on the random angles with uniform distribution

a network are normalized to increase the generalization power (Miyato et al. 2018; Salimans and Kingma 2016). However, normalization schemes decrease the speed of learning.

5.2.4 Activation Function Family

The activation function is a part of a neural network offering non-linearity power to the network. If all activation functions are linear, the entire network is just a linear model. Thus, activation functions have great roles in the non-linearity of the networks. In shallow networks, *sigmoid* or *tanh* are usually applied. In deep networks, the exploiting and vanishing of gradient (Bengio et al. 1994) are common problems. For solving vanishing gradient, Rectified Linear Units (ReLU) is utilized as the activation function (Nair et al. 2010). The derivative of this function is greater than one when the corresponding neuron is active. In Glorot and Bengio (2010), the role of activation function on the learning performance of deep models has been discussed. The extensions of this activation function have been also used for regularization, see (Clevert et al. 2016; Xu et al. 2015; He et al. 2015; Jie et al. 2020).

5.3 Conclusions and Future Works on the Active Schemes

Since active methods impose a dynamic noise on the learning model or learning algorithm, the model does not learn the training data's noises. A model that learns the training patterns' noise fails to perform well on the testing dataset (Salman and Liu 2019). To investigate the effect of different noise types on the training process, consider an example on the Yale dataset (Lee et al. 2005), including 165 images from 15 classes. The input size is $32 * 32$. To train a multilayer perceptron (MLP) neural network, we consider the following six scenarios:

- (S1): Training on data set without noise and regularization,
- (S2): Training on data set without noise and with Dropout regularization,
- (S3): Training on data set with fixed noise and without regularization,
- (S4): Training on data set with fixed noise and Dropout regularization,
- (S5): Training on data set with fixed noise and applying PCA on data for noise smoothing and without regularization,
- S6: Training on data set with fixed noise and using PCA on data for noise smoothing and Dropout regularization.

Figure 7 compares the results of the training and testing errors for these scenarios. As we expected, the worst case happens when the learning model tries to learn the noises (S3). The (S5) scenario also converse rapidly, similar to (S3), while its testing error is better than (S3). This result shows that smoothing the noise by PCA causes an improvement in the testing error. Comparing (S3) and (S5) with (S1) shows that the imposing noise accelerates the training process, while it cannot solve overfitting completely. The best testing result is obtained from (S2) that trains with regularization. The results of (S6) and (S4) are also close to (S2). It shows that the active methods can rarely remove the effect of data noise.

In Table 7, the results of active methods are compared, considering the learning components. The following results can be summarized:

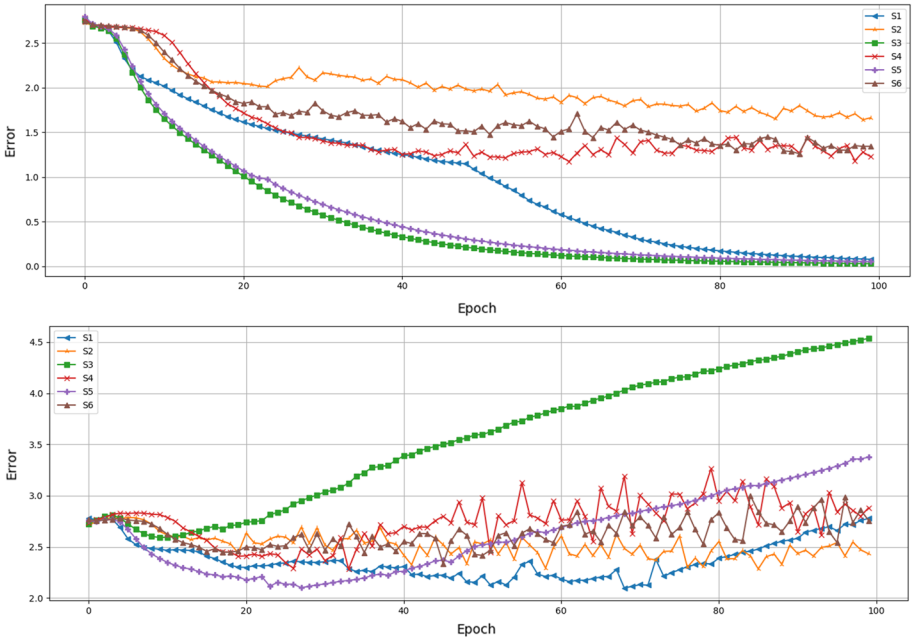

Fig. 7 Comparison between training and testing errors (in the top and down sub-figures) by considering the noise on samples and training process

- They increase the training time but probably less than passive methods do.
- If the imposing noise is excessive, the model learns slowly, while if the noise is limited, the model is likely to overfit. To show evidence, consider a VGG-16 network to train CIFAR-10 data with different p for Dropout regularization. Let p belongs to $\{0.9, 0.3, 0.01\}$. Figure 8 presents the training results. Since $p = 0.9$ is very high, a tremendous amount of noise is imposed on the learning model, and the convergence speed is meager. $p = 0.01$ is very small, and it seldom affects the training process, and the testing loss increases after some iterations. Finally, $p = 0.3$ performs well to control overfitting. However, in the final steps, the testing error rises slightly. As a result, this experiment shows that regularization hyper-parameters are fundamental, and the training is highly dependent on these values.
- The level of the overfitting changes in each training iteration and the magnitude of noise should be adapted.
- Typically, the active regularization methods do not adapt their parameters with respect to the level of overfitting. For an instance of an adaptable version, see (Bejani and Ghatee 2020a) which uses a small parameter for overfitting control in the first steps of the training, as the model is underfitted. Conversely, it increases the regularization parameters when $v(t)$ becomes large.
- The combination of data augmentation and adaptive regularization can also be investigated in future studies.

Table 7 The regularization schemes for neural networks (Active methods with imposing noise on learning algorithms)

Refs.	Name	Affects	Description
Srivastava et al. (2014)	Dropout		By a probability distribution, in each iteration, some neurons are chosen for training and the others are left
Wan et al. (2013)	Dropconnect		Similar to dropout, selects some random weights from different neurons to train
Bejani and Ghatee (2020a)	Adaptive Dropout		A dropout regularization whose parameter adapts with the overfitting statue
Kang et al. (2018)	Shakeout	Directly on Weights	Changes the weights by a piecewise function of a special random variable
Khan et al. (2018)	Bridgeout		The weights are perturbed in each training iteration based on a a Bernoulli matrix
Krueger et al. (2017)	Zoneout		A special dropout for recurrent neural networks
Larsson et al. (2017)	Drop-Path		A version of dropout for CNNs with block architecture to train blocks randomly
Ma et al. (2020)	Dropout with Tabu		A version of dropout that adds a diversification strategy by marking the dropped units
Bejani and Ghatee (2020b)	Adaptive SVD Regularization (ASR)		Approximates network weights with their SVD, when the validation error exceeds from training error substantially
Bejani and Ghatee (2020)	Adaptive LRF		For each layer, the overfitting is evaluated separately and when it is high, the layer simplifies by getting low-rank matrix factorization (LRF)
Khan et al. (2019)	Spectral Dropout	Output of Layers	In each layer, the output is computed based on the spectral of the frequency transformation and convolution filters
Antoniou et al. (2017)	DAGAN	Input Data	Uses a Generative Adversarial Network (GAN) with encoder, decoder, and discriminator. The encoder output is combined with a normal distribution vector and is passed to a decoder to generate fake data (augmented data). The discriminator classifies the fake data and the real data
Wu and He (2018)	Group Normalization	Input of Layers	Similar to batch-normalization works but normalizes some data channels
Salimans and Kingma (2016)	Weight Normalization	Post-process on the Weights	Normalizes the weights of a network in each training iteration

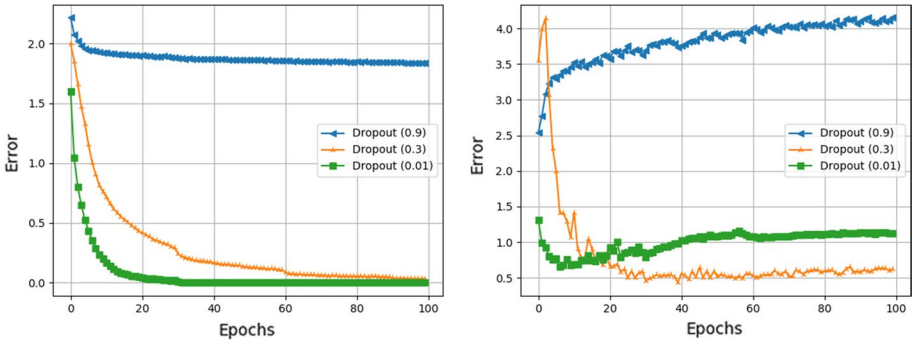

Fig. 8 Comparison between training and testing errors (in the top and down sub-figures) by considering different p for Dropout regularization method

6 Overfitting Control by Semi-active Methods

These methods do not affect the entire architecture of the network and do not train several models simultaneously. They update networks by adding or removing neurons or connections. They are designed with bottom-up and top-down paradigms. The methods that add neurons or connections are called bottom-up, while other methods that remove neurons or connections are called top-down.

6.1 Pruning Method (Top-Down)

To simplify a model that learns the samples but drops in overfitting, the pruning procedure removes the unnecessary parameters (Han et al. 2015). Han et al. (2015) prunes a network in three steps. The first step is training the network and finding essential connections. The second step removes all links whose weights are less than a predefined threshold. In the last step, the network is trained again to retrieve the performance. It uses different regularization schemes, including weight decay or dropout, to improve the results. The dropout parameters are adjustable for each layer in the retraining step by the following:

$$D_r = D_o \sqrt{\frac{C_{ir}}{C_{io}}}, \tag{14}$$

where D_o is initialized randomly. C_{ir} and C_{io} are the numbers of connections of the ith layer before and after pruning.

Hu et al. (2016) introduces a network trimming method for deep networks that iteratively prunes unimportant neurons based on output analysis. For a convolution layer, Averages Percentage of Zeros (APoZ) is evaluated as the following:

$$APoZ_c^{(i)} = \frac{\sum_{n=1}^{N} \sum_{m=1}^{M} f(O_{c,m}^{(i)}(n) = 0)}{N \times M}, \tag{15}$$

where $O_c^{(i)}$ is the output of the cth channel in ith layer. $f(x) = 1$ if x is true, and $f(x) = 0$ otherwise. N is the number of validation samples, and M is the dimension of output $O_c^{(i)}$.

When $APoZ_c^{(i)}$ is great, cth channel in ith layer is eliminated. Then, the network is trained again. These steps repeat while the validation error can improve.

Matrix computation methods are also useful to prune the network and to avoid overfitting. For example, (Hassibi et al. 1993) uses the inverse of the Hessian matrix to prune the network. Schittenkopf et al. (1997), applies principal components to reduce the dimension of both inputs and internal representations, causing better generalization and less overfitting. There are different criteria to detect the connections for pruning. Cottrell et al. (1994) removes the weights when they have low relevancy. The details of the pruning schemes are summarized in Table 8. For more study, one can also see (Reed 1993).

6.2 Network Construction Method (Bottom-Up)

Assume a simple network that cannot learn the samples. Adding new parameters, including neurons or a hidden layer, the network complexity increases, and the new network is retrained to improve the results. This iterative method can be executed several times.

Moody and Antsaklis (1996) uses a dependency identification algorithm to add a hidden layer when the performance is poor. In each iteration, if the error of the trained network is less than a pre-defined threshold, the following optimization problem is solved for all data-batch $i \in B$ to find the weights vector w_i:

$$\min_{w_i} trace((\boldsymbol{\Phi}(w_i u) - d)^T(\boldsymbol{\Phi}(w_i u) - d)), \tag{16}$$

where m is the batch-size, n is the output feature, $u \in \mathbb{R}^{m\times n}$ is the output of the last layer, $\boldsymbol{\Phi}()$ is an activation function, and d is the desired output. The function $trace()$ returns the trace of the matrix. After solving this optimization problem, matrix W containing columns w_i is considered as the weights of a new hidden layer.

Fahlman and Lebiere (1990) adds a hidden neuron that maximizes the following sum of correlations over candidate neurons $o \in O$ and samples $p \in P$:

$$o = arg - max\left\{\sum_{o\in O}\left|\sum_{p\in P}(V_{p,o} - \bar{V}_o)(E_{p,o} - \bar{E}_o)\right|\right\}, \tag{17}$$

where $V_{p,o}$ is the output of neuron o for the sample p, $E_{p,o}$ is the corresponding residual error, and $\bar{V}_o$ and $\bar{E}_o$ are the means of $V_{p,o}$ and $E_{p,o}$ over all samples.

Setiono (2001) uses the cross-validation and trains a small network. If adding a new hidden neuron improves accuracy, it continues. Otherwise, it comes back to the former network. Similarly, to add multiple hidden nodes, consider two networks, including H and $H + h$ hidden neurons. If the accuracy of the first network overcomes the second one, the algorithm terminates. Otherwise, it trains a network with $H + 2h$ hidden neurons. This process continues while the accuracy can improve. To avoid overfitting during the training of these networks, one can use the following regularization term (Setiono 1997):

$$R(w, v) = \epsilon_1\left(\sum_{i,j}\frac{\beta w_{i,j}^2}{1+\beta w_{i,j}^2} + \sum_{i,j}\frac{\beta v_{i,j}^2}{1+\beta v_{i,j}^2}\right) + \epsilon_2\left(\sum_{i,j} w_{i,j}^2 + \sum_{i,j} v_{i,j}^2\right), \tag{18}$$

where ϵ_1, ϵ_2, and β are positive coefficients, w is the weight matrix between the input layer and the hidden layer, and v is the weight matrix between the hidden layer and the output layer.

Table 8 The pruning schemes for neural networks simplification (Semi-active methods)

Refs.	Name	Description
Yu et al. (2018)	NISP	Uses the integer programming problem $\min_{s_j} \sum_{i=1}^{N} \langle s_j, \lvert F_l(x_i) - F_l(s_j \odot x_i) \rvert \rangle$, to prune the network, where s_j is a binary vector indicating pruned neurons, F_l is the output of lth layer, and N is the number of training samples
Anwar et al. (2017)	Structure Pruning	States two heuristic functions for selecting and for pruning their connections with the least information lost. The first heuristic uses a genetic algorithm. The second one is an activation sum voting similar to max-pooling. It computes the average of the outputs of the neurons on the validation dataset and prunes the neurons whose outputs tend to zero
He et al. (2017)	Channel Pruning	Prunes the network in each convolution layer by solving $\min_{\beta,W} \frac{1}{2N} \lVert Y - \sum_{i=1}^{c} \beta_i X_i W_i^T \rVert_F^2 + \lambda \lVert \beta \rVert_1$ where $\lVert \beta \rVert_0 \leq c'$, and c' is a positive parameter less than the number of channels in the convolution layer. Y is the output of the layer. X_i and W_i are the inputs and the weights of ith channel of a convolution layer that meet $\lVert W_i \rVert_F = 1$
Hassibi et al. (1993)	OBS with Hessian Matrix	The pruning is done in an iterative procedure by combining Optimal Brain Surgeon (OBS) LeCun et al. (1990) with Hessian Matrix
Dong et al. (2017)	Layer-wise OBS	The idea is similar to Hassibi et al. (1993) and prunes the network by solving a minimization perturbed problem
Abbasi et al. (2013)	RBF Units Pruning	Fuzzy resource allocating network (FRAN) with fuzzy outputs adds a new RBF unit when the training accuracy is poor and prunes any inactive neuron
Han et al. (2015)	Connection Removing	The pruning algorithm is done in three steps: 1-training the network, 2-removing unimportant connections less than a threshold, 3- retraining the network. When all connections to a neuron are removed, the neuron removes
Hu et al. (2016)	Network Trimming	For network pruning APoZ measure is defined to measures the percentage of zero activations of a neuron after ReLU mapping
Schittenkopf et al. (1997)	PCA based Pruning	Using the Principle Components Analysis (PCA) on the outputs of the layers, this method removes the redundant neurons
Cottrell et al. (1994)	Statistical Method	In each iteration, a statistical test is used to find less important weights and to set zero
Tian and Pan (2020)	Enhanced Transfer Function	Reduces the less important weights of the process and improves the network feature extraction capability
Luo et al. (2017)	Thinet (Thinner Net)	For pruning the filters of convolutional neural networks, the properties of layer $l+1$ is used by a greedy algorithm to prune the filters of layer l
Liu et al. (2014)	Pruning with Hessian Diagonal	Similar to Hassibi et al. (1993), but the non-diagonal elements of the Hessian matrix of the loss function are assumed as zero

Table 8 (continued)

Refs.	Name	Description
He et al. (2017)	Reconstruct Pruning	For CNNs, the pruning is done in two steps: LASSO regression-based channel selection and least square reconstruction
Mitsuno et al. (2020)	Structured Sparse Regularization	For CNNs, filter-wise grouping, neuron-wise grouping, feature-wise grouping, and hierarchical grouping are defined to prune the unnecessary subsets of weights

Zhang et al. (2014) constructs a neural network architecture by a two-stage orthogonal least-squares method. It determines model terms, model coefficients, and model size by minimizing the following Akaike Information Criterion (AIC):

$$AIC = N \log\left(\frac{1}{N}\langle E(M), E(M)\rangle\right) + 2M_s, \tag{19}$$

where $E(M)$ is the residual error of model M, $\langle ., . \rangle$ is the inner product, N is the size of the dataset, and M_s is the model size. Let model M^* minimizes AIC. To determine the necessary terms of M^* in each step, one can minimize the Error Reduction Ratio (ERR). Based on the ERR, a term that maximally reduces the error is selected.

The bottom-up methods for network construction are compared based on their termination criteria and the construction methods in Table 9.

6.3 Conclusions and Future Works on the Semi-active Schemes

The construction methods suffer from some problems. Firstly, they use the validation dataset to construct the next layer or the next block. Secondly, there are few efforts to construct deep neural networks for a specific task. Meanwhile, to construct a network, a subset of network parameters can be optimized, but it is an NP-hard problem (He et al. 2017). Usually, network construction is used for simple tasks and shallow networks.

On the other hand, compression techniques simplify the network (Serra et al. 2020; Yang et al. 2020). They belong to the semi-active methods, but they are somewhat different. Many parameters of a deep network cause some difficulties in computing the output, especially on mobile devices with power limitation (Bejani and Ghatee 2020a). To solve this problem, Moblie-Net uses depth-wise convolution, and so the number of operations decreases, while its accuracy remains acceptable (Howard et al. 2017; Sandler et al. 2018). There are four major approaches to compressing the deep networks, including parameter pruning and sharing, low-rank factorization, compact convolutional filters, and knowledge distillation. In this regard, (Cheng et al. 2018) reviews different compression methods. They seek and eliminate unnecessary details of deep models. Thus, the models resulting from compression schemes are smoother, and then the corresponding models are regularized and can avoid overfitting. However, compression and overfitting controlling methods are different and play different roles. Thus, they cannot replace each other. The compression methods should decrease the complexity of the model and may lower the model performance. Conversely, the semi-active methods do not disturb the model performance and improve generalization power. The following conclusions on semi-active methods are useful:

- The computation time of semi-active methods is long. But they overcome passive methods.
- Using statistical tests for semi-active methods improves their performance.
- Semi-active methods are widely applicable in the shallow networks, including RBF networks, Extreme Learning Machines (ELM), and other feed-forward networks.
- Semi-active methods can construct the networks or prune them in parallel.

However, the application of semi-active methods in deep networks is challenging. For deep network construction, trial and error schemes are tedious, and usually, expertise plays a more critical role. Possibly, a recommender system enables to collect such expertness to

Table 9 The construction schemes for neural networks designing (Semi-active methods)

Refs.	Termination criterion	Unit construction
Moody and Antsaklis (1996)	Comparing the training performance with a pre-defined threshold	Adding a new hidden layer whose weights are solution of (16)
Fahlman and Lebiere (1990)	Comparing the residual error with a pre-defined threshold	Adding a new hidden neuron with the maximum correlation between its output and the residual error (17)
Setiono (2001)	Not an improvement in the accuracy of the learning model	Several hidden neurons simultaneously together and using regularization term
Zhang et al. (2014)	Comparing the model size with an upper bound M_s and finding the minimum of the residual error	Model terms, model coefficients, and model size
Narasimha et al. (2008)	Comparing the number of added neurons with an upper bound	Adding a random number of the neurons in each step and training by OWO-HWO Chen et al. (1999). In 20% of epochs, it trains the new neurons; for the rest, it trains all neurons
Zemouri and An evolutionary building algorithm for deep neural networks, in, (2017)	Comparing the loss value with a pre-defined threshold, the maximum number of layers and neurons with two upper bounds	Adding new neurons or layers

generate new semi-active methods to control overfitting in deep networks. For example, (Abpeikar et al. 2020) develops an expert node in the neural tree to prevent overfitting. Similarly, one can create a new block in deep learning models to prune or re-construct the network.

7 Lessons from Literature

The overfitting occurs in shallow and deep networks, but it is not easy to control its effect. Obviously, this phenomenon affects deep networks more broadly. To descibe the effect of overfitting controllers in deep networks, an example is given by Bejani and Ghatee (2020b) on GPS dataset where the differences of the training accuracy and testing accuracy for different regularization schemes including ASR, early stopping, dropout, dropconnect, shakeout, spectral dropout, bridgeout, noise injection, and weight decay are 11.2, 13.1, −2.6, 18.1, 4.7, 0.4, −0.5, 14.9 and 17.1, respectively. This difference for the learning model without regularization is 15.6. This example proves the necessity of defining an effective overfitting controller for any learning task. On the other hand, using some general measures such as complexity, efficiency, convergence speed, and scalability in different networks, the effectiveness of each overfitting controller can be evaluated. Tables 10, 11 compare these measures. Generally, we conclude the following points.

Lesson 1: Overfitting controllers usually decrease the convergence speed of the training algorithms. For online learning, the methods proposed in Clevert et al. (2016), Ba et al. (2016), Ioffe and Szegedy (2015), Bejani and Ghatee (2020b), Bejani and Ghatee (2020a) are useful. The low-speed methods can only be used for off-line applications on very complex datasets. For better understanding, the processing times of different learning models with and without overfitting control can be compared. Unfortunately, many references have not reported such a comparison. To quantitatively measure the processing time, the number of FLoating-point OPeration (FLOP) can be considered. A lower FLOP indicates the superiority of the model and low computation complexity. Generally speaking, the overfitting controllers do not cause a heavy computational overhead. Meanwhile, in some cases, such as pruning methods (Han et al. 2015; Vu et al. 2019), the computational burden can be reduced, and definitely, the accuracy can slightly diminish. Table 12 reports more information on FLOPs.

Lesson 2: To study the effect of overfitting controllers on the accuracy, their results on the famous standard image datasets are analyzed. Table 13 presents the top-1 error on CIFAR-10, CIFAR-100, SVHN and MNIST, together with the top-5 error on ImageNet. Although not all of the references have evaluated the same analysis, the reported results are sufficiently insightful. By comparing these results with baseline networks without overfitting control, auto-augmentation provides results with minimum errors. Also, when augmentation and other kinds of overfitting controllers are used together, the generalization power grows significantly (Bejani and Ghatee 2020a). To reach a fair comparison, the Average of Relative Improvement (ARI) is defined as follows:

$$ARI = \frac{\sum_{d \in DataSets} max\{(Error_{WC} - Error_{OC}), 0\}/Error_{WC}}{|DataSets|} \tag{20}$$

where $|DataSets|$ shows the number of datasets, $Error_{WC}$ and $Error_{OC}$ indicate the errors of the models without and with the overfitting control. Based on the last column of Table 13,

Table 10 A general comparison between different methods to control overfitting

Refs.	Complexity	Efficiency	Convergence speed			Scalability
			Increasing	Decreasing	Extremely decreasing	
Bengio (2000)	Between $O(H)$ and $O(N^2)$	Possibly overfits on the validation dataset	–	✓	–	N/A
Snoek et al. (2012)	$O(IN)$		–	–	✓	
Suganuma et al. (2017)	$O(PIN)$, P: the genetic population size		–	–	✓	
Zoph and Le (2017)	$O(CPIN)$, P: number of sampled networks, C: training complexity of LSTM Sak et al. (2014)		–	–	✓	
Bejani and Ghatee (2020b)	$O(I(\sum_{l=1}^{L} n_l^3 + N))$, n_l: max-dimension of weight matrix of layer l, L: number of layers	Automatically decreases the overfitting	✓	–	–	Most of the deep networks
Tomar and Rose (2014)	$O(T^3 + IN)$	Related to Tikhonov coefficient	–	✓	–	CNN
Ayinde et al. (2019)	$O(N^2)$	Diverse features are produced by great Thikonov coefficient. When the noise of the input data is great, it takes the noise as a unique feature and cannot control overfitting	–	✓	–	ResNet He et al. (2016), DenseNet Huang et al. (2017)
Srivastava et al. (2014)	$O(N)$, without redundant computation	The lower dropout probability, the more noise is imposed on the gradient. It controls overfitting	–	✓	–	Most of the deep networks
Bejani and Ghatee (2020a)		The influence of the dropout or the weight decay changes with respect to the training states	✓	–	–	CNN

Table 10 (continued)

Refs.	Complexity	Efficiency	Convergence speed			Scalability
			Increasing	Decreasing	Extremely decreasing	
Zhong et al. (2020)	$O(N+B)$, B: batch-size	Considers new parts of the input data, but cannot decrease the model complexity	–	✓	–	ResNet He et al. (2016), Wide-ResNet Zagoruyko and Komodakis (2016)
Ma et al. (2019)	$O(N)$	By increasing the coefficient, the redundant weights can be removed	–	✓	–	CNN
Cubuk et al. (2019)	$O(C+IN)$	Generates new policies, when the validation error does not decrease. It causes overfitting on the validation data	–	–	✓	ResNet He et al. (2016)

N, H, I, C, and T are the numbers of parameters, hyper-parameters, iteration, LSTM cells, and training samples, respectively

Table 11 Continue of Table 10

Ref.	Complexity	Efficiency	Convergence Speed			Scalability
			Increasing	Decreasing	Extremely decreasing	
Ioffe and Szegedy (2015)	$O(BL+N)$, L: number of batch-normalization layers	By increasing training parameters, the model complexity increases, and it causes overfitting	✓	–	–	Inception Szegedy et al. (2015)
Ba et al. (2016)	$O(FL+N)$, F: number of the features		✓	–	–	CNN
Salimans and Kingma (2016)	$O(N)$	The normalization of the weights in each iteration limits the active neurons and the model complexity decreases	–	✓	–	CNN
Clevert et al. (2016)	$O(N)$	When the activation of the neurons increases, it searches the space better and faster. It is useful for low overfitting	✓	–	–	CNN
Bejani and Ghatee (2020a)	O(R), R : selected regularization scheme	The overfitting solves when the model complexity is more than the data complexity	✓	–	–	CNN
Crammer et al. (2013)	$O(N^2)$	Since network weights are sampled with normal distribution, the overfitting is probably low	–	✓	–	N/A

N, H, I, C, and T are the numbers of parameters, hyper-parameters, iteration, LSTM cells, and training samples, respectively

Table 12 Comparison between FLOP of the different schemes with and without overfitting control on different datasets

Ref.	Experimental results
Ma et al. (2019)	Integrated transformed ℓ_1 does not drastically reduce accuracy, while reduces FLOPs significantly. It decreases FLOPS on MNIST, Fashion-MNIST, SDD, Pendigits, CIFAR-10 and CIFAR-100 datasets to 10.7%, 30.9%, 15.8%, 33.0%, 28.3% and 58.9% of the corresponding FLOPs of ℓ_2 models
Bejani and Ghatee (2020b)	Compares the FLOPs of Wide-Resnet-28 $*$ 10 with ASR, Weight Decay, and without regularization. They are 11333.31, 11333.29, and 11332.98 GFLOPS, respectively. Thus ASR and weight decay just increases the FLOP counts 0.0029% and 0.0027%, respectively
Louizos et al. (2018)	The expected FLOPs during training for ℓ_0 regularized networks are less than the original dropout. On CIFAR-10 and CIFAR-100, the FLOPs are approved. On these datasets, the expected FLOP belongs to $[3.25, 3.3] * 10^{11}$ while expected FLOP for dropout is close to $3.5 * 10^{11}$
Vu et al. (2019)	On different ResNet models with a constant FLOP count for the training, it is shown that adding mixup (cutout) regularization improves accuracy 0.18−1.37% (1.04−2.13%) compared with the baseline. Also, adding pruning decreases accuracy between 0.7and1.07% and decreases FLOPs between 14.6and15.2%
Han et al. (2015)	Using ℓ_1 regularization to penalize non-zero parameters after pruning and before retraining, FLOP counts reduce 8%, 16%, 30% and 21% for Lenet-300-100, Lenet-5, AlexNet, and VGG-16, respectively. On the ImageNet dataset, the number of parameters of AlexNet reduces from 61 to 6.7 million, and the number of parameters of VGG-16 reduces from 138 to 10.3 million with no loss of accuracy

the overfitting controllers can be ranked. As can be seen, dropout (Srivastava et al. 2014), adaptive dropout (Bejani and Ghatee 2020a), adaptive weight decay (Bejani and Ghatee 2020a), auto-augmentation (Cubuk et al. 2019) and enhancing diversity of features (Ayinde et al. 2019) have shown the best results.

Lesson 3: In addition to the accuracy, overfitting controllers can be compared by other machine learning measures, including precision, recall, and f-measure (Powers 2011). Such comparisons on regularized deep networks are limited. Sometimes, this gap leads to misunderstandings. Table 14 outlines the comparisons found. Some references, including figures of these measures (Li et al. 2016), as well as the references without the baseline results (Shekar and Dagnew 2020), have been excluded from this table. Note that since the datasets are different, the results are not comparable. However, the methods with great precision and recall are the best candidates to follow in general learning tasks.

Lesson 4: There is no priority among active and passive schemes to improve training performance. Nevertheless, active methods are almost less time-consuming than passive methods. Sometimes, an active method provides better accuracy and computational time. For example, dropout (Srivastava et al. 2014) does not add more computational burden, while its performance is better than meta-heuristic algorithms (Suganuma et al. 2017).

Lesson 5: When the computational capacity and the training time are not limited, passive methods can be used. However, the resulting model may overfit on the validation data. Random sampling can help solve this problem.

Lesson 6: When the training time is limited, semi-active methods are suggested for shallow networks.

Table 13 Comparison between percent of errors of the different schemes with and without overfitting control on famous image datasets

Refs.	Overfitting control	Top-1 Error				Top-5 Error	ARI
		CIFAR-10	CIFAR-100	SVHN	MNIST	ImageNet	
Cubuk et al. (2019)	Without Overfit Cont.	2.7	14.0	1.4	–	3.9	26.25
	Auto-Augmentation	**1.5**	**10.7**	**1.0**	–	**3.5**	
Ayinde et al. (2019)	Without Overfit Cont.	13.63			1.4	8.56	26.66
	Enhancing Diversity of Features	6.30			1.10	8.0	
Zoph and Le (2017)	Without Overfit Cont.	3.46					0
	NAS	3.65					
Snoek et al. (2012)	Without Overfit Cont.	17.98					16.68
	Bayesian Optimization	14.98					
Snoek et al. (2015)	Without Overfit Cont.	7.25	33.71				15
	Bayesian Optimization	6.37	27.4				
Clevert et al. (2016)	Without Overfit Cont.	4.5	27.62			–	6
	ELU	6.55	24.28			Below 10%	
Mitsuno et al. (2020)	Without Overfit Cont.	24.50	42.80		0.97		0.09
	Structured Sparse Regularization	23.98	40.60		0.75		
Tam and Dunson (2020)	Without Overfit Cont.	71.12			9.75		14.61
	Fiedler	47.74			3.9		
Bejani and Ghatee (2020b)	Without Overfit Cont.	11.9	48.0	4.2	1.6		17.95
	ASR	11.8	47.3	3.9	**0.6**	–	
Bejani and Ghatee (2020b)	Just Augmentation	7.8	44.6	5.5			20.13
	ASR+Augmentation	6.8	44.4	2.9	–	–	
Bejani and Ghatee (2020)	Without Overfit Cont.	28.2		2.8			19.4
	Adaptive LRF	24.3		2.1			
Bejani and Ghatee (2020)	Just Augmentation	7.0		2.4			13.35
	Adaptive LRF+Augmentation	6.0		2.1			

Table 13 (continued)

Refs.	Overfitting control	Top-1 Error				Top-5 Error	ARI
		CIFAR-10	CIFAR-100	SVHN	MNIST	ImageNet	
Srivastava et al. (2014)	Without Overfit Cont.	15.60	43.48	3.95	1.6	26.2	31.38
	Dropout	12.61	37.20	2.55	0.79	16.4	
Ma et al. (2019)	Without Overfit Cont.	22.82	29.94		1.11	50.03	1.7
	Integrated Transform ℓ_1	21.96	29.52		1.50	50.18	
Suganuma et al. (2017)	Without Overfit Cont.	24.1					2.61
	Meta-Heuristic Algorithm	23.47					
Zhao et al. (2019)	Without Overfit Cont.	≈ 34			≈ 1		≈ 2.9
	Adaptive regularization	≈ 32			≈ 1		
Bejani and Ghatee (2020a)	Without Overfit Cont.	44		18			
	Adaptive dropout	40		9			29.5
	Adaptive weight decay	39		10			27.85

The best results are highlighted in bold face

Table 14 Comparison between different regularization schemes on deep models based on precision, recall, F-measure

Ref.	Dataset	Regularization Method	Precision	Recall	F-measure
Nurhaida et al. (2020)	Batik	Without Overfit Cont.	0.61	0.65	0.62
		Batch normalization	0.82	0.83	0.82
		Batch normalization after tuning parameters	0.85	0.85	0.85
Pashaei et al. (2019)	Accident images analysis dataset	Without overfit Cont. for Accident detection	0.90	0.90	0.90
		Ensembling for accident detection	0.99	0.99	0.99
		Without overfit Cont. for severity classification	0.67	0.48	0.49
		Ensembling for severity classification	0.90	0.69	0.72
		Without Overfit Cont. for vehicles classification	0.75	0.74	0.74
		Ensembling for vehicles Classification	0.92	0.92	0.92
Ahmed et al. (2020)	Snapshot Serengeti	Without Overfit Cont.	0.69	0.62	0.64
		Noise injection to labels (Average on 30%)	0.75	0.65	0.67
	Panama-Netherlands	Without Overfit Cont.	0.49	0.48	0.46
		Noise injection to labels (Average on 30%)	0.59	0.57	0.54
Xu et al. (2016)	Neural morphology images	Without Overfit Cont.	0.75	0.82	0.81
		Semi-supervised Graph-smooth Regularization	0.90	0.83	0.86
Tao et al. (2018)	MNIST	Without Overfit Cont.	97.30	98.58	97.93
		Regularized Deep Embedded Clustering	98.74	99.58	99.16
Alani et al. (2018)	LSP	ℓ_2 Regularization	96	96	96
		ℓ_2 Regularization+Data Augmentation	1	1	1

Table 14 (continued)

Ref.	Dataset	Regularization Method	Precision	Recall	F-measure
Bejani and Ghatee (2020a)	SVHN	Without Overfit Cont.	0.81	0.80	0.81
		Adaptive Dropout	0.91	0.90	0.90
		Adaptive Weight Decay	0.90	0.88	0.89
	CIFAR-10	Without Overfit Cont.	0.57	0.56	0.56
		Adaptive Dropout	0.60	0.60	0.60
		Adaptive Weight Decay	0.61	0.61	0.61
	STL	Without Overfit Cont.	0.27	0.28	0.27
		Adaptive Dropout	0.31	0.29	0.29
		Adaptive Weight Decay	0.32	0.31	0.30

Lesson 7: In cases with limited computational capacity or limited training time, the active methods are the best options. Meanwhile, their hyper-parameters can be adjusted by an expert system; see (Abpeikar et al. 2020) for an initial plan.

Lesson 8: Sometimes, regularization methods decrease the generalization power, as they make mistake in choosing the model with the right complexity due to the data complexity. In these cases, regularization is destructive. For example, in Bejani and Ghatee (2020b), there is an experiment on MobileNet without regularization where the training accuracy is 100%, and the testing accuracy is 84%. When the regularization methods, including early stopping, dropout, dropconnect, shakeout, spectral dropout, and noise injection, have been used, the testing accuracies decrease to 65.1%, 39.2%, 80.1%, 79.4%, and 75.3%. In these situations, the tuning of the regularization parameters with respect to the data complexity is essentially required.

Lesson 9: The training procedure is related to a nonlinear optimization problem that can be transformed into a system of linear or nonlinear equations. When these systems are ill-posed (Tikhonov and Arsenin 1977), the solution is unstable, and the corresponding learning model becomes overfitted (Bejani and Ghatee 2020b). Using condition number, it is possible to detect an ill-posed component of the system. Through causality analysis, once a component is discovered as the reason for the ill-posed problem, only this component can be corrected to improve the system performance. Possibly, this type of causality analysis can be furthered to tune-up overfitting controllers of deep models in future works.

8 Conclusion

Learning models deal with unseen data prediction based on training on finite seen data. Sometimes the seen data are not adequate to give a fair inference. When a learning model fits these data, the model cannot learn some real data details. Although the model learns seen data entirely, it cannot generalize on unseen data. It is entitled to overfitting. There are three branches of schemes to control overfitting; passive, active, and semi-active methods. They affect the different parts of the learning model, training algorithm, or training data. In this paper, we classified the most important schemes to control overfitting in these branches. We compare their concepts, criteria, advantages, and disadvantages for each of them and extract their lessons. We also stated some future works for each branch. This systematic review helps find a roadmap for the next research and select a reasonable overfitting controller for different shallow and deep neural network models.

Acknowledgements The authors would like to sincerely appreciate the anonymous reviewers for their leading comments in three rounds.

References

Abbasi E, Ghatee M, Shiri ME (2013) FRAN and RBF-PSO as two components of a hyper framework to recognize protein folds. Comput Biol Med 43(9):1182–1191

Abbasi E, Shiri ME, Ghatee M (2016) A regularized root-quartic mixture of experts for complex classification problems. Knowl-Based Syst 110:98–109

Abpeikar S, Ghatee M, Foresti GL, Micheloni C (2020) Adaptive neural tree exploiting expert nodes to classify high-dimensional data. Neural Netw 124:20–38

Abpeykar S, Ghatee M (2019) Neural trees with peer-to-peer and server-to-client knowledge transferring models for high-dimensional data classification. Expert Syst Appl 137:281–291

Abpeykar S, Ghatee M (2019) An ensemble of RBF neural networks in decision tree structure with knowledge transferring to accelerate multi-classification. Neural Comput Appl 31(11):7131–7151
Abpeykar S, Ghatee M, Zare H (2019) Ensemble decision forest of RBF networks via hybrid feature clustering approach for high-dimensional data classification. Comput Stat Data Anal 131:12–36
Ahmed A, Yousif H, Kays R, He Z (2020) Animal species classification using deep neural networks with noise labels. Ecol Inform 57:101063
Alani AA, Cosma G, Taherkhani A, McGinnity T (2018) Hand gesture recognition using an adapted convolutional neural network with data augmentation. In: 4th international conference on information management (ICIM). IEEE, pp 5–12
Amer M, Maul T (2019) A review of modularization techniques in artificial neural networks. Artif Intell Rev 52(1):527–561
Antoniou A, Storkey A, Edwards H (2017) Data augmentation generative adversarial networks, 1–14. arXiv:1711.04340
Anwar S, Hwang K, Sung W (2017) Structured pruning of deep convolutional neural networks. ACM J Emerg Technol Comput Syst 13(3):1–18
Arpit D, Zhou Y, Kota BU, Govindaraju V (2016) Normalization propagation: a parametric technique for removing internal covariate shift in deep networks. In: Proceedings of the 33rd international conference on international conference on machine learning (ICML). JMLR, pp 1168–1176
Asadi AR, Abbe E (2020) Chaining meets chain rule: multilevel entropic regularization and training of neural networks. J Mach Learn Res 21(139):1–32
Ayinde BO, Inanc T, Zurada JM (2019) Regularizing deep neural networks by enhancing diversity in feature extraction. IEEE Trans Neural Netw Learn Syst 30(9):2650–2661
Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. In: Advances in neural information processing systems (NIPS)-deep learning symposium, 1–14
Beck A, Ben-Tal A (2006) On the solution of the tikhonov regularization of the total least squares problem. SIAM J Optim 17(1):98–118
Bejani MM, Ghatee M (2019) Regularized deep networks in intelligent transportation systems: a taxonomy and a case study, 1–8. arXiv:1911.03010
Bejani MM, Ghatee M (2021) Least auxiliary loss-functions with impact growth adaptation (laliga) for convolutional neural networks. Neurocomputing (in press)
Bejani MM, Ghatee M (2020) Adaptive low-rank factorization to regularize shallow and deep neural networks. arXiv:2005.01995
Bejani MM, Ghatee M (2018) A context aware system for driving style evaluation by an ensemble learning on smartphone sensors data. Transp Res Part C Emerg Technol 89:303–320
Bejani MM, Ghatee M (2020) Convolutional neural network with adaptive regularization to classify driving styles on smartphones. IEEE Trans Intell Transp Syst 21(2):543–552
Bejani MM, Ghatee M (2020) Theory of adaptive SVD regularization for deep neural networks. Neural Netw 128:33–46
Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396
Belkin M, Niyogi P, Sindhwani V (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res 7(Nov):2399–2434
Belkin M, Niyogi P (2002) Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems (NIPS), pp 585–591
Bengio Y (2000) Gradient-based optimization of hyperparameters. Neural Comput 12(8):1889–1900
Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157–166
Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(Feb):281–305
Bishop CM (1995) Training with noise is equivalent to tikhonov regularization. Neural Comput 7(1):108–116
Blanc G, Gupta N, Valiant G, Valiant P (2020) Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In: Conference on learning Theory, pp 483–513
Calvetti D, Reichel L (2003) Tikhonov regularization of large linear problems. BIT Numer Math 43(2):263–283
Caruana R, Lawrence S, Giles CL (2001) Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: Advances in neural information processing systems (NIPS), pp 402–408
Cawley GC, Talbot NL (2007) Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. J Mach Learn Res 8(Apr):841–861

Cawley GC, Talbot NL (2010) On over-fitting in model selection and subsequent selection bias in performance evaluation. J Mach Learn Res 11(Jul):2079–2107
Chen H, Yao X (2009) Regularized negative correlation learning for neural network ensembles. IEEE Trans Neural Netw 20(12):1962–1979
Chen H-H, Manry MT, Chandrasekaran H (1999) A neural network training algorithm utilizing multiple sets of linear equations. Neurocomputing 25(1–3):55–72
Cheng Y, Wang D, Zhou P, Zhang T (2018) Model compression and acceleration for deep neural networks: the principles, progress, and challenges. IEEE Signal Process Mag 35(1):126–136
Chollet F et al (2015) Keras. https://keras.io
Clevert D-A, Unterthiner T, Hochreiter S (2016) Fast and accurate deep network learning by exponential linear units. In: 4th international conference on learning representations (ICLR), 1–14
Cottrell M, Girard B, Girard Y, Mangeas M, Muller C (1994) SSM: a statistical stepwise method for weight elimination. In: International conference on artificial neural networks. Springer, pp 681–684
Crammer K, Kulesza A, Dredze M (2013) Adaptive regularization of weight vectors. Mach Learn 91(2):155–187
Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV (2019) Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 113–123
Dai W, Yang Q, Xue G-R, Yu Y (2007) Boosting for transfer learning. In: Proceedings of the 24th international conference on machine learning. ACM, pp 193–200
Darwish A, Hassanien AE, Das S (2020) A survey of swarm and evolutionary computing approaches for deep learning. Artif Intell Rev 53(3):1767–1812
Demyanov S (2015) Regularization methods for neural networks and related models, Ph.D. thesis, Department of Computing and Information System, The University of Melbourne
Dietterich T (1995) Overfitting and undercomputing in machine learning. ACM Comput Surv (CSUR) 27(3):326–327
Ding S, Li H, Su C, Yu J, Jin F (2013) Evolutionary artificial neural networks: a review. Artif Intell Rev 39(3):251–260
Donahue J, Krähenbühl P, Darrell T(2017) Adversarial feature learning. In: 5th international conference on learning representations (ICLR), 1–18
Dong X, Chen S, Pan S (2017) Learning to prune deep neural networks via layer-wise optimal brain surgeon. In: Advances in neural information processing systems (NIPS), pp 4857–4867
Eftekhari HR, Ghatee M (2018) Hybrid of discrete wavelet transform and adaptive neuro fuzzy inference system for overall driving behavior recognition. Transp Res Part F Traffic Psychol Behav 58:782–796
Eigenmann R, Nossek JA (1999) Gradient based adaptive regularization. In: Proceedings of the IEEE signal processing society workshop on neural networks for signal processing. IEEE, pp 87–94
Eldén L (2019) Matrix methods in data mining and pattern recognition, vol 15. SIAM, Bangkok
Engelbrecht AP (2001) A new pruning heuristic based on variance analysis of sensitivity information. IEEE Trans Neural Netw 12(6):1386–1399
Erhan D, Bengio Y, Courville A, Manzagol P-A, Vincent P, Bengio S (2010) Why does unsupervised pretraining help deep learning? J Mach Learn Res 11(Feb):625–660
Fahlman SE, Lebiere C (1990) The cascade-correlation learning architecture. In: Advances in neural information processing systems (NIPS), pp 524–532
Feurer M, Hutter F (2019) Hyperparameter optimization. In: Hutter F, Kotthoff L, Vanschoren J (eds) Automated machine learning. The springer series on challenges in machine learning. Springer, Cham, pp 3–33. https://doi.org/10.1007/978-3-030-05318-5_1
Finnoff W, Hergert F, Zimmermann H-G (1993) Extended regularization methods for nonconvergent model selection. In: Advances in neural information processing systems (NIPS), pp 228–235
Fong S, Deb S, Yang X-S (2018) How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics. In: Sa P, Sahoo M, Murugappan M, Wu Y, Majhi B (eds) Progress in intelligent computing techniques: theory, practice, and applications. Advances in intelligent systems and computing, vol 518. Springer, Singapore, pp 3–25. https://doi.org/10.1007/978-981-10-3373-5_1
Franceschi L, Donini M, Frasconi P, Pontil M (2017) Forward and reverse gradient-based hyperparameter optimization. In: Proceeding of the 34th international conference on machine learning-volume 70. JMLR, pp 1165–1173
Frank L, Hubert E (1996) Pretopological approach for supervised learning. In: Proceedings of 13th international conference on pattern recognition, vol 4. IEEE, pp 256–260
Galdran A, Alvarez-Gila A, Meyer MI, Saratxaga CL, Araújo T, Garrote E, Aresta G, Costa P, Mendonça AM, Campilho A (2017) Data-driven color augmentation techniques for deep skin image analysis, 1–4. arXiv:1703.03702

Geman S, Bienenstock E, Doursat R (1992) Neural networks and the bias/variance dilemma. Neural Comput 4(1):1–58
Girosi F, Jones M, Poggio T (1995) Regularization theory and neural networks architectures. Neural Comput 7(2):219–269
Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th international conference on artificial intelligence and statistics, pp 249–256
Golub GH, Hansen PC, O'Leary DP (1999) Tikhonov regularization and total least squares. SIAM J Matrix Anal Appl 21(1):185–194
Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems (NIPS), pp 2672–2680
Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3(Mar):1157–1182
Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. In: Advances in neural information processing systems (NIPS), pp 1135–1143
Harvey N, Liaw C, Mehrabian A (2017) Nearly-tight vc-dimension bounds for piecewise linear neural networks. In: Conference on learning theory, pp 1064–1068
Hassibi B, Stork DG, Wolff GJ (1993) Optimal brain surgeon and general network pruning. In: IEEE international conference on neural networks. IEEE, pp 293–299
Hawkins DM (2004) The problem of overfitting. J Chem Inf Comput Sci 44(1):1–12
Heidari M, Ghatee M, Nickabadi A, Nezhad AP (2020) Diverse and styled image captioning using SVD-based mixture of recurrent experts, 1–13. arXiv:2007.03338
He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034
He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision, pp 1389–1397
Ho TK, Basu M, Law MHC (2006) Measures of geometrical complexity in classification problems. In: Data complexity in pattern recognition. Springer, pp 1–23
Ho TK, Baird HS (1998) Pattern classification with compact distribution maps. Comput Vis Image Understand 70(1):101–110
Ho TK, Basu M (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 3:289–300
Hoekstra A, Duin RP (1996) On the nonlinearity of pattern classifiers. In: Proceedings of 13th international conference on pattern recognition, vol 4. IEEE, pp 271–275
Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications, 1–9. arXiv:1704.04861
Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
Hu H, Peng R, Tai Y-W, Tang C-K (2016) Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. arXiv:1607.03250
Ioffe S (2017) Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In: Advances in neural information processing systems (NIPS), pp 1945–1953
Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd international conference on international conference on machine learning—volume 37. JMLR, pp 448–456
Ivanova I, Kubat M (1995) Initialization of neural networks by means of decision trees. Knowl-Based Syst 8(6):333–344
Jaafra Y, Laurent JL, Deruyver A, Naceur MS (2019) Reinforcement learning for neural architecture search: a review. Image Vis Comput 89:57–66
Jeng J-T (2005) Hybrid approach of selecting hyperparameters of support vector machine for regression. IEEE Trans Syst Man Cybern Part B (Cybern) 36(3):699–709
Jie R, Gao J, Vasnev A, Tran M-n (2020) Regularized flexible activation function combinations for deep neural networks, 1–12. arXiv:2007.13101
Jin J, Dundar A, Culurciello E (2015) Robust convolutional neural networks under adversarial noise, 1–8. arXiv:1511.06306
Kang G, Li J, Tao D (2018) Shakeout: a new approach to regularized deep neural network training. IEEE Trans Pattern Anal Mach Intell 40(5):1245–1258

Khan N, Shah J, Stavness I (2018) Bridgeout: stochastic bridge regularization for deep neural networks. IEEE Access 6:42961–42970
Khan SH, Hayat M, Porikli F (2019) Regularization of deep neural networks with spectral dropout. Neural Netw 110:82–90
Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations (ICLR), pp 1–14
Kirsch L, Kunze J, Barber D (2018) Modular networks: learning to decompose neural computation. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31 (NeurIPS 2018), pp 2408–2418. https://papers.nips.cc/paper/2018
Krogh A, Hertz JA (1991) A simple weight decay can improve generalization. In: Moody J, Hanson S, Lippmann RP (eds) Advances in neural information processing systems 4 (NIPS 1991), pp 950–957. https://papers.nips.cc/paper/1991
Krueger D, Maharaj T, Kramár J, Pezeshki M, Ballas N, Ke NR, Goyal A, Bengio Y, Courville A, Pal C (2017) Zoneout: regularizing RNNs by randomly preserving hidden activations. In: 5th international conference on learning representations (ICLR), 1–11
Kwasigroch A, Mikołajczyk A, Grochowski M (2017) Deep convolutional neural networks as a decision support tool in medical problems-malignant melanoma case study. In: Polish control conference. Springer, pp 848–856
Larsen J, Hansen LK, Svarer C, Ohlsson M (1996) Design and regularization of neural networks: the optimal use of a validation set. In: Proceedings of the IEEE signal processing society workshop on neural networks for signal processing. IEEE, pp 62–71
Larsen J, Svarer C, Andersen LN, Hansen LK (2012) Adaptive regularization in neural network modeling. In: Montavon G, Orr GB, Müller KR (eds) Neural networks: tricks of the trade. Lecture notes in computer science, vol 7700. Springer, Berlin, Heidelberg, pp 111–130. https://doi.org/10.1007/978-3-642-35289-8_8
Larsson G, Maire M, Shakhnarovich G (2017) Fractalnet: ultra-deep neural networks without residuals. In: 5th international conference on learning representations (ICLR), 1–11
Lawrence S, Giles CL, Tsoi AC (1997) Lessons in neural network training: Overfitting may be harder than expected. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence. AAAI Press, pp 540–545
LeCun YA, Bottou L, Orr GB, Müller K-R (2012) Efficient backprop. In: Neural networks: tricks of the trade. Springer, pp 9–48
LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Advances in neural information processing systems (NIPS), pp 598–605
Lee K-C, Ho J, Kriegman DJ (2005) Acquiring linear subspaces for face recognition under variable lighting. IEEE Trans Pattern Anal Mach Intell 27(5):684–698
Leung K-C, Leung CH (2011) Improvement of fingerprint retrieval by a statistical classifier. IEEE Trans Inf Forensics Secur 6(1):59–69
Li X, Zhao L, Wei L, Yang M-H, Wu F, Zhuang Y, Ling H, Wang J (2016) Deepsaliency: multi-task deep neural network model for salient object detection. IEEE Trans Image Process 25(8):3919–3930
Li H, Liu D, Wang D (2017) Manifold regularized reinforcement learning. IEEE Trans Neural Netw Lear Syst 29(4):932–943
Li H, Wang X, Ding S (2018) Research and development of neural network ensembles: a survey. Artif Intell Rev 49(4):455–479
Li Y, Liu F (2016) Whiteout: Gaussian adaptive noise regularization in deep neural networks, 1–17. arXiv:1612.01490
Li H, Liu D, Wang D (2015) Approximate policy iteration with unsupervised feature learning based on manifold regularization. In: 2015 international joint conference on neural networks (IJCNN). IEEE, pp 1–6
Li F, Tian C, Zuo W, Zhang L, Yang M-H (2018) Learning spatial-temporal regularized correlation filters for visual tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4904–4913
Liu Y, Wu W, Fan Q, Yang D, Wang J (2014) A modified gradient learning algorithm with smoothing l1/2 regularization for takagi-sugeno fuzzy models. Neurocomputing 138:229–237
Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26
Liu X, Fan F, Kong L, Diao Z, Xie W, Lu J, You J (2020) Unimodal regularized neuron stick-breaking for ordinal classification. Neurocomputing 388:34–44
Liu Z, Castagna J (1999) Avoiding overfitting caused by noise using a uniform training mode. In: International joint conference on neural networks (IJCNN), vol 3. IEEE, pp 1788–1793

Liu R, Liu E, Yang J, Li M, Wang F (2006) Optimizing the hyper-parameters for SVM by combining evolution strategies with a grid search. In: Huang DS, Li K, Irwin GW (eds) Intelligent control and automation. Lecture notes in control and information sciences, vol 344. Springer, Berlin, Heidelberg, pp 712–721. https://doi.org/10.1007/978-3-540-37256-1_87
Liu C, Zhang Z, Wang D (2014) Pruning deep neural networks by optimal brain damage. In: 15th annual conference of the international speech communication association
Louizos C, Welling M, Kingma DP (2018) Learning sparse neural networks through L_0 regularization. In: 6th international conference on learning representations (ICLR), 1–13
Luo J-H, Wu J, Lin W (2017) Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision, pp 5058–5066
Luo P, Wang X, Shao W, Peng Z (2019) Towards understanding regularization in batch normalization. In: 7th international conference on learning representation (ICLR), 1–23
Ma R, Miao J, Niu L, Zhang P (2019) Transformed l_1-regularization for learning sparse deep neural networks. Neural Netw 119:286–298
Ma Z, Sattar A, Zhou J, Chen Q, Su K (2020) Dropout with tabu strategy for regularizing deep neural networks. Comput J 63(7):1031–1038
Maaten Lvd, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9(Nov):2579–2605
Maclaurin D, Duvenaud D, Adams R (2015) Gradient-based hyperparameter optimization through reversible learning. In: International conference on machine learning, pp 2113–2122
Martín-Félez R, Xiang T (2014) Uncooperative gait recognition by learning to rank. Pattern Recogn 47(12):3793–3806
Miche Y, Van Heeswijk M, Bas P, Simula O, Lendasse A (2011) TROP-ELM: a double-regularized ELM using LARS and tikhonov regularization. Neurocomputing 74(16):2413–2421
Mikołajczyk A, Grochowski M (2018) Data augmentation for improving deep learning in image classification problem. In: International interdisciplinary PhD workshop (IIPhDW). IEEE, pp 117–122
Mirza M, Osindero S (2014) Conditional generative adversarial nets, 1–7. arXiv:1411.1784
Mitsuno K, Miyao J, Kurita T (2020) Hierarchical group sparse regularization for deep convolutional neural networks, 1–8. arXiv:2004.04394
Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for generative adversarial networks. In: 6th international conference on learning representations (ICLR), pp 1–26
Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning. arXiv:1312.5602
Mockus J, Tiesis V, Zilinskas A (1978) The application of Bayesian methods for seeking the extremum. Towards Global Optim 2:117–129
Moher D, Liberati A, Tetzlaff J, Altman DG (2009) Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. Ann Intern Med 151(4):264–269
Monari G, Dreyfus G (2002) Local overfitting control via leverages. Neural Comput 14(6):1481–1506
Moody JO, Antsaklis PJ (1996) The dependence identification neural network construction algorithm. IEEE Trans Neural Netw 7(1):3–15
Nabian MA, Meidani H (2020) Physics-driven regularization of deep neural networks for enhanced engineering design and analysis. J Comput Inf Sci Eng 20(1):011006, Paper No: JCISE-19-1072. https://doi.org/10.1115/1.4044507
Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML), pp 807–814
Nannen V (2003) The paradox of overfitting, Ph.D. thesis, Faculty of Artificial Intelligence, the University of Groningen
Narasimha PL, Delashmit WH, Manry MT, Li J, Maldonado F (2008) An integrated growing-pruning method for feedforward network training. Neurocomputing 71(13–15):2831–2847
NarasingaRao M, Prasad VV, Teja PS, Zindavali M, Reddy OP (2018) A survey on prevention of overfitting in convolution neural networks using machine learning techniques. Int J Eng Technol (UAE) 7(2.32):177–180
Nashed MZ (1986) The theory of tikhonov regularization for fredholm equations of the first kind (c. w. groetsch). SIAM Rev 28(1):116–118
Natterer F (1984) Error bounds for tikhonov regularization in Hilbert scales. Appl Anal 18(1–2):29–37
Ng AY (1997) Preventing" overfitting" of cross-validation data. In: Proceedings of the 14th international conference on machine learning (ICML), vol 97. Citeseer, pp 245–253
Nowlan SJ, Hinton GE (1992) Simplifying neural networks by soft weight-sharing. Neural Comput 4(4):473–493

Nurhaida I, Ayumi V, Fitrianah D, Zen RA, Noprisson H, Wei H (2020) Implementation of deep neural networks (DNN) with batch normalization for batik pattern recognition. Int J Electr Comput Eng (2088–8708) 10:2045–2053
Pan Z, Yu W, Yi X, Khan A, Yuan F, Zheng Y (2019) Recent progress on generative adversarial networks (GANs): a survey. IEEE Access 7:36322–36333
Pashaei A, Ghatee M, Sajedi H (2019) Convolution neural network joint with mixture of extreme learning machines for feature extraction and classification of accident images. J Real-Time Image Proc 17:1051–1066
Patel V, Mujumdar N, Balasubramanian P, Marvaniya S, Mittal A (2019) Data augmentation using part analysis for shape classification. In: IEEE winter conference on applications of computer vision (WACV). IEEE, pp 1223–1232
Perez L, Wang J (2017) The effectiveness of data augmentation in image classification using deep learning, 1–8. arXiv:1712.04621
Phan DN, Le Thi HA (2019) Group variable selection via $\ell_{p,0}$ regularization and application to optimal scoring. Neural Netw 118:220–234
Powers DM (2011) Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation. J Mach Learn Technol 2(1):37–63
Real E, Aggarwal A, Huang Y, Le QV (2019) Regularized evolution for image classifier architecture search. Proc AAAI Conf Artif Intell 33(01):4780–4789
Reed R (1993) Pruning algorithms-a survey. IEEE Trans Neural Netw 4(5):740–747
Ren M, Zeng W, Yang B, Urtasun R (2018) Learning to reweight examples for robust deep learning. In: 35th international conference on machine learning, 1–13
Reunanen J (2003) Overfitting in making comparisons between variable selection methods. J Mach Learn Res 3(Mar):1371–1382
Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326
Sak H, Senior A, Beaufays F (2014) Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: 15th annual conference of the international speech communication association
Salamon J, Bello JP (2017) Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Signal Process Lett 24(3):279–283
Salimans T, Kingma DP (2016) Weight normalization: a simple reparameterization to accelerate training of deep neural networks. In: Advances in neural information processing systems (NIPS), pp 901–909
Salman S, Liu X (2019) Overfitting mechanism and avoidance in deep neural networks, 1–8. arXiv :1901.06566
Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520
Sarle WS (1995) Stopped training and other remedies for overfitting. In: Proceedings of the 27th symposium on the interface of computing science and statistics, pp 352–360
Schaffer JD, Whitley D, Eshelman LJ (1992) Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: International workshop on combinations of genetic algorithms and neural networks. IEEE, pp 1–37
Scherzer O, Engl HW, Kunisch K (1993) Optimal a posteriori parameter choice for tikhonov regularization for solving nonlinear ill-posed problems. SIAM J Numer Anal 30(6):1796–1838
Schittenkopf C, Deco G, Brauer W (1997) Two strategies to avoid overfitting in feedforward networks. Neural Netw 10(3):505–516
Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
Serra T, Kumar A, Ramalingam S (2020) Lossless compression of deep neural networks, 1–14. arXiv :2001.00218
Setiono R (1997) A penalty-function approach for pruning feedforward neural networks. Neural Comput 9(1):185–204
Setiono R (2001) Feedforward neural network construction using cross validation. Neural Comput 13(12):2865–2877
Sharma A, Zaidi A, Singh R, Jain S, Sahoo A (2013) Optimization of svm classifier using firefly algorithm. In: 2013 IEEE second international conference on image information processing (ICIIP-2013). IEEE, pp 198–202
Shekar B, Dagnew G (2020) L1-regulated feature selection and classification of microarray cancer data using deep learning. In: Proceedings of 3rd international conference on computer vision and image processing. Springer, pp 227–242

Smith FW (1968) Pattern classifier design by linear programming. IEEE Trans Comput 100(4):367–372
Smith SP, Jain AK (1988) A test to determine the multivariate normality of a data set. IEEE Trans Pattern Anal Mach Intell 10(5):757–761
Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in neural information processing systems (NIPS). Curran Associates Inc, New York, pp 2951–2959
Snoek J, Rippel O, Swersky K, Kiros R, Satish N, Sundaram N, Patwary M, Prabhat M, Adams R (2015) Scalable Bayesian optimization using deep neural networks. In: International conference on machine learning, pp 2171–2180
Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127
Suganuma M, Shirakawa S, Nagao T (2017) A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the genetic and evolutionary computation conference. ACM, pp 497–504
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 conference on computer vision and pattern recognition, pp 1–9
Tam E, Dunson D (2020) Fiedler regularization: Learning neural networks with graph sparsity, 1–10. arXiv:2003.00992
Tao Y, Takagi K, Nakata K (2018) RDEC: integrating regularization into deep embedded clustering for imbalanced datasets. Proc Mach Learn Resear 95:1–16
Taylor L, Nitschke G (2018) Improving deep learning with generic data augmentation. In: IEEE symposium series on computational intelligence (SSCI), pp 1542–1547
Tian Y, Pan G (2020) An unsupervised regularization and dropout based deep neural network and its application for thermal error prediction. Appl Sci 10(8):2870
Tikhonov A, Arsenin VY (1977) Methods for solving ill-posed problems. Wiley, Hoboken
Tomar VS, Rose RC (2014) Manifold regularized deep neural networks. In: 15th annual conference of the international speech communication association, pp 1–5
Tzafestas S, Dalianis P, Anthopoulos G (1996) On the overtraining phenomenon of backpropagation neural networks. Math Comput Simul 40(5–6):507–521
Vapnik VN, Chervonenkis AY (2015) On the uniform convergence of relative frequencies of events to their probabilities. In: Vovk V, Papadopoulos H, Gammerman A (eds) Measures of complexity. Springer, Cham, pp 11–30. https://doi.org/10.1007/978-3-319-21852-6_3
Vapnik V (2006) Estimation of dependences based on empirical data. Springer, Berlin
Vauhkonen M, Vadasz D, Karjalainen PA, Somersalo E, Kaipio JP (1998) Tikhonov regularization and prior information in electrical impedance tomography. IEEE Trans Med Imag 17(2):285–293
Verma V, Lamb A, Beckham C, Najafi A, Mitliagkas I, Lopez-Paz D, Bengio Y (2019) Manifold mixup: better representations by interpolating hidden states. In: Proceedings of the 36th international conference on machine learning, vol 97. PMLR, pp 6438–6447
Verma V, Qu M, Lamb A, Bengio Y, Kannala J, Tang J (2019) Graphmix: regularized training of graph neural networks for semi-supervised learning, 1–16. arXiv:1909.11715
Vu T, Wen E, Nehoran R (2019) How not to give a FLOP: combining regularization and pruning for efficient inference. Comput Sci Res 197:1–14
Wang S, Li D, Song X, Wei Y, Li H (2011) A feature selection method based on improved fisher's discriminant ratio for text sentiment classification. Expert Syst Appl 38(7):8696–8702
Wan L, Zeiler M, Zhang S, Le Cun Y, Fergus R (2013) Regularization of neural networks using dropconnect. In: International conference on machine learning, pp 1058–1066
Wąsowicz M, Grochowski M, Kulka M, Mikołajczyk A, Ficek M, Karpieńko K, Cićkiewicz M (2017) Computed aided system for separation and classification of the abnormal erythrocytes in human blood. In: Spigulis J (ed) Biophotonics-Riga 2017, vol 10592. International Society for Optics and Photonics, SPIE, pp 49–55
Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8(3–4):229–256
Wittek P (2014) Quantum machine learning: what quantum computing means to data mining. Academic Press, Cambridge
Wu W, Fan Q, Zurada JM, Wang J, Yang D, Liu Y (2014) Batch gradient method with smoothing l1/2 regularization for training of feedforward neural networks. Neural Netw 50:72–78

Wu Y, He K (2018) Group normalization. In: Proceedings of the European conference on computer vision (ECCV), pp 3–19
Xu K, Su H, Zhu J, Guan J-S, Zhang B (2016) Neuron segmentation based on CNN with semi-supervised regularization. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 20–28
Xu B, Wang N, Chen T, Li M (2015) Empirical evaluation of rectified activations in convolutional network. In: Deep learning workshop, ICML, pp 1–5
Yam JY, Chow TW (2000) A weight initialization method for improving training speed in feedforward neural network. Neurocomputing 30(1–4):219–232
Yang H, Tang M, Wen W, Yan F, Hu D, Li A, Li H, Chen Y (2020) Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 678–679
Yang T, Zhu S, Chen C (2020)GradAug: a new regularization method for deep neural networks. arXiv :2006.07989
Yu Y, Gong Z, Zhong P, Shan J (2017) Unsupervised representation learning with deep convolutional neural network for remote sensing images. In: International conference on image and graphics. Springer, pp 97–108
Yu R, Li A, Chen C-F, Lai J-H, Morariu VI, Han X, Gao M, Lin C-Y, Davis LS (2018) Nisp: pruning networks using neuron importance score propagation. In: The IEEE conference on computer vision and pattern recognition (CVPR)
Zagoruyko S, Komodakis N (2016) Wide residual networks, 1–15. arXiv:1605.07146
Zemouri R (2017) An evolutionary building algorithm for deep neural networks. In: 12th international workshop on self-organizing maps and learning vector quantization. Clustering and data visualization (WSOM) 2017:1–7. https://doi.org/10.1109/WSOM.2017.8020002
Zhang L, Li K, Bai E-W, Irwin GW (2014) Two-stage orthogonal least squares methods for neural network construction. IEEE Trans Neural Netw Learn Syst 26(8):1608–1621
Zhang Q, Yang LT, Chen Z, Li P (2018) A survey on deep learning for big data. Inf Fusion 42:146–157
Zhao H, Tsai Y-HH, Salakhutdinov RR, Gordon GJ (2019) Learning neural networks with adaptive regularization. In: Wallach H, Larochelle H, Beygelzimer A, Alché-Buc Fd, Fox E, Garnett R (eds) Advances in neural information processing systems (NIPS). Curran Associates, Inc., New York, pp 11393–11404
Zhao M, Tang B, Deng L, Pecht M (2020) Multiple wavelet regularized deep residual networks for fault diagnosis. Measurement 152:107331
Zhong Z, Zheng L, Kang G, Li S, Yang Y (2020) Random erasing data augmentation. In: Proceedings of the AAAI conference on artificial intelligence (AAAI)
Zhu X, Zhou W, Li H (2018) Improving deep neural network sparsity through decorrelation regularization. In: Proceedings of the 27th international joint conference on artificial intelligence (IJCAI), pp 3264–3270
Zoph B, Le QV (2017) Neural architecture search with reinforcement learning. In: 5th international conference on learning representations (ICLR), pp 1–16
Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J R Stat Soc Ser B (Stat Methodol) 67(2):301–320