$R_e = \{r_{e_1}, ..., r_{e_i}, ..., r_{e_n}\}$ – is the set of rules possessed by an expert system of type $r_{e_i} : p_{e_i}; a_{e_i} \to b_{e_i}; n_{e_i}$, where $a_{ei}(S_e)$ – is the existence condition for inference b_{ei}, moreover S_e includes S_v ($S_v = \{S_{v_1}, S_{p_{v2}}, ..., S_{v_i}, ..., S_{v_n}\}$), $S_v \subset S_a$ ($S_v \subset S_e$), b_i – are the results of redefinition of S_e, which again can result in redefinition of S_r. As far as S_r can cause change of the user's behavior (this changes environment influencing of the agent's behavior), which, again is observed by the expert system; videlicet behavior of user at a moment of time t_1 (beginning of the expert system operation) results in changes of distributed network resource at a moment in time t_2 and in possible subsequent behavior of user at a moment in time t_3 and so on (Пд – behavior):

$$S_r^{t_n} = \left(S_r^{t_{n-1}}, S_e^{t_{n-1}} \Пд^{t_{n-1}} \left(S_r^{t_{n-2}} \left(S_r^{t_{n-3}}, S_e^{t_{n-3}}, \Пд^{t_{n-3}} \times \right.\right.\right.$$
$$\left.\left.\left. \times \left(S_r^{t_{n-4}} \left(... \left(S_r^{t_2}, S_e^{t_2} \Пд^{t_2} \left(S_r^{t_1} \right) \right) \right) \right) \right) \right).$$

A model of interaction between distributed network resource and users has now been developed. Its main peculiarity is the use of the multi-agent expert system. Methods of agent management allow the rationalization of the present interaction by means of the user's behavior reflexive control via dynamic modifications in the representation and the content of network resource.

On the base of educational the following resource (www.i5nfo.ru) the present methods are applicable for tasks of interaction between organizations, distributed network resources, and the user. They have also been applied to instruments measuring and monitoring radiation levels, which had been affirmed by implementation acts.

References

1. Aripova O. Models of Interaction between User and Distributed Network Resource: Research and information magazine. Innovations. SPb. : JSC "TRANSFER", 2009.

2. Andreychikov A., Andreychikova O. Intelligent Information Systems : the Textbook. M. : Finance and statistics, 2006. P. 424.

3. Gavrilova T., Khoroshevskiy V. Intelligent Systems Knowledgebases. SPb. : Piter, 2001.

4. Gushin A. Basic Concepts of Personal-Centered Information Systems Development / Voenmeh. Baltic State Technical University Bull. SPb. : "Sot" printing establishment, 2008. P. 34–44.

5. Laurier J.-L. Artificial Intelligence Systems: transl. from the French. M. : Mir, 1991. P. 568.

6. Rassel S., Norwig P. Artificial Intelligence: Modern Approach. M. : Williams, 2006. P. 1048.

7. Yasnitskiy L. Artificial Intelligence Guide-book : Study guide for students of inst. of tertiary education. M. : Academia, 2005. P. 176.

V. V. Bukhtoyarov, E. S. Semenkin
Siberian State Aerospace University named after academician M. F. Reshetnev, Russia, Krasnoyarsk

A COMPREHENSIVE EVOLUTIONARY APPROACH FOR NEURAL NETWORK ENSEMBLES AUTOMATIC DESIGN

A new comprehensive approach for neural network ensembles design is proposed. It consists of a method of neural networks automatic design and a method of automatic formation of an ensemble solution on the basis of separate neural networks solutions. It is demonstrated that the proposed approach is not less effective than a number of other approaches for neural network ensembles design.

Keywords: neural networks, ensemble, automatic design, genetic programming, probabilistic evolutionary algorithm.

At the present time data analysis systems which are based on intelligent information technologies are increasingly demanded in many fields of human activity and the scale requirements to these systems are continuously increasing. In connection with these facts the problem of developing methods for automatic design and adaptation of IIT for specific tasks is becoming more urgent. Such methods could allow to abandon the use of expensive, mostly human, resources for the design of the IIT and to reduce the time required for the development of intelligent systems.

One of the most widely used and popular intellectual technologies are artificial neural networks. The range of problems solved by using neural networks is extremely large because of many advantages of systems based on their use. Despite the fact that this information technology could be called a universal tool for solving problems of data analysis, in each case we have to create a unique neural network. One of the approaches to improve the efficiency of systems based on the use of neural networks is the use of neural networks ensembles. Problem solving with the help of neural network ensembles supposes

simultaneous use of a finite number of preliminarily trained neural networks.

This approach was first proposed in [1], where it has been shown that the ability of a neural network system to generalize can be significantly improved by uniting a number of component neural networks into an ensemble.

In general, neural network ensemble formation consists of two steps. For the use in data analysis systems it is desirable to automate both of these steps. The first step is to form the structure and to train a number of component networks which will be included in an ensemble or a preliminary pool. The second step is to select the networks, the solutions of which will produce a final solution, and to define a way and parameters of a common solution formation.

The approach based on preliminary choice and fixation of neural networks' structure and their training on different learning sets formed from an original learning sample is often used to perform the first stage of a neural networks ensemble formation. In some cases in the process of training of each subsequent neural network special attention is paid to its training on those subsets of a learning set on which the mistake of all previous neural networks was big. Many other approaches for forming and training of component networks are also widespread. Neural networks with different number of hidden neurons are offered to use in work [2]. Usage of different object functions for training of each component network is advised in work [3]. The main disadvantage of such approaches is the need for a priori fixed structure of component networks which could adversely affect the adaptability of the resulting collective solution.

Other promising approaches developed in different works are based on the use of genetic algorithms (GA) [4] to form component neural networks. The main difficulty in these approaches is the necessity to tune a great number of GA parameters. Besides, it is often very difficult to choose proper settings for GA, because it requires a lot of time and computational efforts. At the same time a poorly tuned GA may fail to solve the task.

The analysis of existing methods for automatic design of neural networks (in particular the method based on the use of genetic algorithms [4]) shows that methods accumulate and process statistical information about the structure of the designed neural network. However this information is not used explicitly. Instead information is collected and processed in an implicit form by operators of a specific method, the type of which depends on the setting of relevant parameters. But if we suggest a method of processing this information in an explicit form, it would allow to avoid tuning a great number of genetic algorithm parameters, which is very difficult in solving real practical problems. In connection with this we propose a new probabilistic method for automatic generation of neural networks structures. This method uses information-processing principles firstly used in the operators of probabilistic genetic algorithm [5].

The proposed method is based on the computation and use of estimated probabilities $p_{i,j}^{k}$, where $i = \overline{1, N_l}$ is the number of a hidden layer, N_l is a maximum number of hidden layers, $j = \overline{1, N_n}$ is a number of a neuron on a hidden layer, N_n is a maximum number of neurons on a hidden layer, $k = \overline{0, N_F}$, where N_F is the cardinality of a set of activation functions which can be used for neural network structure formation. If $k \in [1; N_F]$ it corresponds to a number of an activation function in a neuron, if $k = 0$ it means the absence of j-th neuron on i-th hidden layer.

A full multi-layer perceptron with the number of hidden layers equal to N_l and the number of neurons in each hidden layer equal to N_n is used as the most complete (in terms of a number of layers and a number of neurons in layers) neural network architecture. This architecture allows to indicate all the possible positions of a neuron in the network clearly with the help of the hidden layer number and numbers of neurons on the layer.

We propose to present solutions as a string S of integers from the interval $[0; N_F]$ with the length equal to $L = N_l \cdot N_n$.

The number of elements of this string is equal to the maximum possible number of neurons on hidden layers of a "full" multilayer perceptron. It is clear that the identification number of each element of the string can be evaluated using the following formula:

$$r = N_n \cdot (i-1) j, \quad i = \overline{1, N_l}, \quad j = \overline{1, N_n} \qquad (1)$$

and it defines the place of a neutron in the structure of an artificial neural network with the architecture of a "full" perceptron.

So each element of the string S can be interpreted in the following way:

– If $S_r = 0$, then j-th neuron on i-th layer does not exist;

– If $S_r \neq 0$, then j-th neuron on i-th layer exists and its activation function identifier is equal to S_r.

A neural network with any configuration of layers and neurons on them, the dimensions of which do not exceed N_l and N_n mentioned for the architecture of a multilayer perceptron, can be described by such a string.

The general scheme of the proposed method for automatic formation of neural networks structure is described below:

1. Execute initialization steps.

2. Form N vectors $S^{o,i}, i = \overline{1, N}$ using random generator, set iteration counter $k = 0$.

3. On k-th step of search to evaluate an objective function for each network presented by $S^{k,i}, i = \overline{1, N}$, establish an intermediate set $S' = \varnothing$.

4. Select N_{par} individuals from a current set $S^k = \{S^{k,1}, S^{k,2}, ..., S^{k,N}\}$ by applying a rank selection procedure and place them into an intermediate set S'.

5. Using the solutions from S' calculate a set of estimated probabilities:

$$\overline{P} = \left\{ p_{i,j}^{l} \right\} = \left\{ p_{r}^{l} \right\}, \quad r = \overline{1, L}, \quad l = \overline{1, N_F} .$$

6. According to the estimated probabilities obtained on the previous step generate a temporary set $\overline{S^k}$, which consists of N new solutions.

7. Apply a mutation operator to each solution of $\overline{S^k}$.

8. Create a new set of solutions S^{k+1} from sets S^k and $\overline{S^k}$.

If a termination criterion is satisfied the best found solution is the result. Otherwise set $k = k + 1$, and go to step 3.

Initialization steps include determination of a maximum number of hidden layers of a neuron network N_l, determination of a maximum number of neurons on each layer of a neuron network N_n, specifying of activation functions set F and a maximum number of steps of N-step structure formation as well as a number of networks looked through at each step of formation of structure N.

The second step implies random generation of a string which represents the initial solutions. In order to generate originally simpler structures of neural networks in this method the following probabilities of the random-number generator for initialization of strings were established:

$$S_r^{1,i} = \begin{cases} 0 \text{ with probability equal to } 0.5 \\ \text{s} \in \left[1; N_F\right] \text{ with probability equal to } 0.5/N_F \end{cases}.$$

Here $r = \overline{1, L}$, $i = \overline{1, N}$. The key steps of our method are step number 5 and step number 6. Considering equation (1) we get the following formula for probabilities evaluation for a fully connected neural network (it is used on step 5):

$$p_i^k = \frac{\sum_{r=N_n(i-1)}^{N_n i} \left| S_r^l : S_r^l = k,\ l = \overline{1, N_{par}} \right|}{N_n \cdot N_{par}},$$

$$p_{i,j}^k = p_i^k, \quad k = \overline{0, N_F}, \quad i = 1, N_l, \quad j = \overline{1, N_n} .$$

Here N_n is a maximum available number of neurons on a hidden layer, N_l is a number of hidden layers, N_{par} is a number of solutions in an intermediate parents' subset, $|\cdot|$ is a set cardinality.

An intermediate set $\overline{S^k}$ is generated according to the probabilities specified above on the sixth step. Accumulation and processing of statistical information about the structures of neural networks in an explicit form are implemented in steps 5 and 6 of the proposed method.

As it was mentioned above the comprehensive approach for neural network ensemble design consists of two main steps. The first step includes design and training of component neural networks. We propose probabilistic approach described above for this step.

The second step includes selection of networks which will be used for collective problem solving and finding out the way according to which the solutions of individuals will be taken into account in a collective solution. The most popular approaches of combining solutions of different neural networks are plurality voting or majority voting for classification problems [1] and simple or weighted averaging for regression problems [6]. The most developed methods are those with weighted averaging and majority voting. For example to evaluate the weighting coefficients of separate neural networks' contributions into the general solution Jimenez [7] uses the evaluation of the quality of their individual solutions. Zhou [8] uses genetic algorithm to find proper weights for each member of an ensemble.

To increase the efficiency of this stage implementation we developed another approach. This method allows you to automatically choose those neural networks that will participate in ensemble solution finding. It also forms an ensemble solution in the form of various transformations and combinations (linear and nonlinear) of individual neural networks solutions. We suppose that, while using the ensemble of neural networks we can find a more efficient solution with the help of formation of more complex combinations of individual neural networks solutions than the simple or weighted averaging and plural or majority voting.

The proposed approach is based on the genetic programming method [9], which is used for solving the tasks of symbolic regression. The basic idea of the proposed method is to adapt and use the operators of genetic programming to automatically generate a solution, which is a formula (a program). This program calculates a general solution of a neural networks ensemble on the basis of solutions received from its individual members.

To devise a solution the proposed method uses the elements of a terminal and a functional sets. Instead of independent variables of the problem (as in genetic programming) answers given by neural networks from a preliminary pool are used as a terminal set T. It includes therms of which solutions will be formed. A terminal set also includes numeric literals tuning of which allows to get a more adaptive general solution of a neural networks ensemble. A functional set F consists of different operations and functions which specify the dependency between an ensemble solution and solutions of individual neural networks.

The general scheme of the proposed method is presented below and basically coincides with the steps of the method of genetic programming used to approximate functions.

1. Initialization of the initial generation.

2. Execute the following steps until the termination criterion is satisfied:

2.1. Calculate fitness for each individual of the current generation.

2.2. With the probability connected with the found value of fitness select one or several individuals for crossover by applying a selection operator.

2.3. Create a new generation of individuals-programs by execution of the following genetic operators:

2.3.1. Create a new individual by applying a crossover operator.

2.3.2. Modify a new individual with a mutation operator.

2.3.3. Copy some initial individual into a new generation.

3. After the termination criterion is satisfied the best found solution is declared as a problem solution.

Solutions are encoded in the form of trees. Outer vertexes are formed from elements of the terminal set and the inside vertexes are formed from the functional elements of the set. Arithmetic operations, mathematical functions, Boolean operations can be used as the functional elements of the set.

An ensemble solution generated by our method is a function input whose parameters are individual solutions of neural networks included in an ensemble:

$$o = f\left(o_1, o_2, ..., o_n\right).$$

Where o is the ensemble solution; o_i is i-th individual neural network solution; n is the number of networks in an ensemble (or a preliminary pool).

The proposed method allows to improve the flexibility of the system based on the use of neural networks ensembles, due to the lack of firmly fixed structure of the interaction between the individual networks. The proposed method not only forms the structure of the interaction between ensemble members, but also indirectly (through inclusion or non-inclusion of the relevant arguments into the formula of general solution) selects those neural network solutions which will be most useful in terms of the solution effectiveness. The use of additional algorithms (e. g. a genetic algorithm) to adjust the parameters of interaction models can further improve the model efficiency.

A number of numerical experiments were carried out on a set of test problems in order to study the effectiveness of the proposed comprehensive approach for neural network ensembles designing. Our approach was compared with three other approaches:

1. GASEN approach [7] – an approach which uses a genetic algorithm to specify weights. Further on while forming an ensemble solution it takes into account those neural networks whose weighting coefficient is higher than some preset value.

2. An approach based on the usage of GA for selection of a fixed number of component networks from a preliminary pool for joint use and specifying their weights (it will be called GA-based 1 below).

3. An approach based on the usage of GA for selection of an arbitrary (non-fixed in advance) number of component networks from a preliminary pool (it will be called GA-based 2 below).

A standard method for neural networks design based on a genetic algorithm with preselected setup of parameters was used to generate a preliminary pool of neural networks in these methods. For neural networks the maximum number of hidden layers was set equal to three, the maximum number of neurons on each layer is equal to five. During the preliminary investigation it was found that the increase of maximum number of hidden layers of the network and the maximum number of neurons on them does not give additional benefits to the formed neural network model for the problem.

To conduct research we also used the data set "Concrete Slump Data Set" from UCI Machine Learning Repository [10; 11]. The data set includes 103 data points describing the dependence between concrete composition and measured figures which characterize deformation and strength of products made of it. There are 7 input variables and 3 output variables in this data set. Examples of other test problems used in comparative studies are listed below (tab. 1).

Table 1

Examples of test problems

Problem	Function	Range of input variables	Sample size
1	$y = \sin x$	$x \in [-4,3]$	150
2	$y = x_1^2 \sin x_1 + x_2^2 \sin x_2$	$x_i \in [-4,3]$	150
3	$y = \frac{x_1 \cdot x_2}{x_3^2}$	$x_i \in [1,20]$	200
4	$y = 100\left(x_2 - x_1^2\right)^2 - \left(1 - x_1\right)^2$	$x_i \in [-2,3]$	200

The main efficiency criterion are estimation of expectation and estimated variance of error calculated after 50 independent runs of algorithms. We use the following formula to calculate an approximation error in each run:

$$\text{Error} = \frac{100\,\%}{s\left(y^{\max} - y^{\min}\right)} \sum_{i=1}^{s} \left|o_i - y_i\right|.$$

Where i is the record number in the sampling; o_i is the output of an ensemble or a single network; y^i is a real value of an output variable in the sampling; $y^{\max}$ and $y^{\min}$ are maximum and minimum values of an output variable, s is a number of instances in the sampling.

The results of a comparative study of the proposed probabilistic method for automatic design of the structure of neural networks and the method using the genetic algorithm (GA) to solve this problem are presented in tab. 2.

The results of a comparative study of the proposed integrated approach for neural networks ensemble design and other methods described above are presented in tab. 3–4.

An example of a formula describing the method of calculating the ensemble solution for Concrete Slump Test problem on the basis of solutions of individual neural networks is given below:

$$o = o_4 - \frac{\left(o_{20} - 1.953 o_4 + o_{18}\right)}{\left(o_9 / \left(o_4 + o_{18}\right) - 2.824\right)},$$

Where o is a solution of a neural networks ensemble; o_4, o_9, o_{18}, o_{20} are escapes of the fourth, ninth, eighteenth and twentieth networks from a pool. These networks were automatically chosen to form an ensemble. Network 18 allows to achieve a minimum error (4.92 %). A maximum error (4.98) corresponds to network 20. An error of a neural networks ensemble is 3.57 %.

Table 2

The results of a comarative study of methods for neural networks automatic design

Problem	Probabilistic method Estimation of an error expectation, %	GA Estimation of an error expectation, %
1	1,880	1,857
2	4,355	4,428
3	0,736	0,780
4	6,750	6,643

Let us analyze the characteristics of a neural networks ensemble design for Concrete Slump Test problem. For comparison we analyze the characteristics of the networks included in the preliminary pool and networks selected for an ensemble. Statistical data were obtained by the results of twenty runs of a software system which implements the proposed method for neural networks automatic design.

There were two hidden layers with two neurons on each layer in the minimum size network included in the preliminary pool. A maximum size network included in the preliminary pool was characterized by the structure of hidden layers 5–4–4, that is, there were 13 neurons on the hidden layers of the network: 5 neurons on the first hidden layer, 4 neurons on the second and third hidden layers. The average number of neurons on the hidden layers of networks included in the preliminary pool is 9.

The average number of networks selected for an ensemble from the 20 networks of a preliminary pool is equal to 5 (the exact value is equal to 4.75). The average number of neurons in such networks is equal to 7.

For the problem of predicting the strength characteristics of concrete the mean of error obtained for neural networks in the preliminary pool is equal to 4,95 %. The estimated mean of error for the averaged output of all the networks from a preliminary pool is equal to 4,58 %. The average error of a neural networks ensemble designed with the proposed method is equal to 3,52 %. It is important to note that an ensemble usually includes not only networks with the lowest individual modeling error but also those neural networks which allow to minimize the total value of errors which characterizes the quality of an ensemble general solution.

The results of statistical studies show that the proposed comprehensive evolutionary approach for neural network ensembles design demonstrates high efficiency in all used test problems. As far as the first phase of the neural network ensemble design based on the results of comparative research is concerned we can conclude that the probabilistic method of forming neural networks structures is not less effective than the commonly used method based on a genetic algorithm. An advantage of this method is the smaller number of algorithm parameters to be set up for its effective work. It is very important for complex tasks due to saving significant time and computational resources necessary to adjust the settings.

The results shows that the proposed approach for neural networks automatic design consisting of a probabilistic method for individual neural networks design and a method for forming an ensemble solution demonstrates high efficiency and it is not inferior to other approaches on the used test problems. The relative superiority of the proposed method over other methods of forming a common solution in neural networks ensembles in terms of an average modeling error is within 50 % for test problem 1 to 13 % per cent for test problem 2. The average superiority of this indicator is about 25 %. For the problem of predicting the concrete strength characteristics relative superiority of the evolutionary method was about 15 %, which can be considered a fairly good result in the processing of real data.

Table 3

Results of comparative study on test problems

Approach	Test problem 1		Test problem 2	
	Estimation of an error expectation, %	Estimated error variance, %	Estimation of an error expectation, %	Estimated error variance, %
Single neural network	1.880	0.510	4.355	0.736
GASEN	1.444	0.112	3.479	0.026
GA based 1	1.335	0.223	3.486	0.028
GA based 2	1.302	0.162	3.482	0.024
Proposed approach	0.855	0.065	3.037	0.003
Approach	Test problem 3		Test problem 4	
	Estimation of an error expectation, %	Estimated error variance, %	Estimation of an error expectation, %	Estimated error variance,%
Single neural network	2.537	0.245	6.643	1.447
GASEN	1.679	0.016	6.192	0.163
GA based 1	1.651	0.019	6.147	0.178
GA based 2	1.639	0.020	6.100	0.209
Proposed approach	1.389	0.035	5.036	0.115

Table 4

Results of comparative study on the problem of concrete strength characteristics prediction

Approach	Estimation of an error expectation, %	Estimated error variance, %
GASEN	4.119	0.040
GA based 1	4.113	0.047
GA based 2	4.012	0.036
Proposed approach	3.521	0.028

The results allow us to draw a conclusion that the proposed approach can effectively automatically generate neural networks ensembles and calculate the final solution of the ensemble. Thus, this integrated approach should be used to improve the effectiveness of the solution of complex practical problems which were traditionally solved by systems using a single artificial neural network, for example, complex problems of approximation and prediction.

References

1. Hansen L. K., Salamon P. Neural network ensembles // IEEE Trans. Pattern Analysis and Machine Intelligence. 1990. № 12 (10). P. 993–1001.

2. Cherkauer K. J. Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks // Proc. AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms : P. Chan, S. Stolfo, D. Wolpert (eds.). Portland, OR : AAAI Press ; Menlo Park, CA. 1996. P. 15–21.

3. Hampshire J., Waibel A. A novel objective function for improved phoneme recognition using timedelay neural networks // IEEE Transactions on Neural Networks. 1990. № 1 (2). P. 216–228.

4. Goldberg D. E. Genetic algorithms in search, optimization and machine learning. Reading. MA: Addison-Wesley, 1989.

5. Semenkin E. S., Sopov E. A. Probabilities-based evolutionary algorithms of complex systems optimization. In Intelligent Systems (AIS'05) and Intelligent CAD (CAD-2005). M. : FIZMATLIT, 2005. Vol. 1. P. 77–79.

6. Perrone M. P., Cooper L. N. When networks disagree: ensemble method for neural networks // Artificial Neural Networks for Speech and Vision ; R. J. Mammone (ed.). N. Y. : Chapman & Hall, 1993. P. 126–142.

7. Jimenez D. Dynamically weighted ensemble neural networks for classification // Proc. IJCNN-98. Vol. 1. Anchorage: AK. IEEE Computer Society Press; Los Alamitos, CA. 1998. P. 753–756.

8. Zhou Z. H., Wu J., Tang W. Ensembling neural networks: Many could be better than all // Artif. Intell. 2002. Vol. 137, № 1–2. P. 239–263.

9. Koza J. R. The Genetic Programming Paradigm: Genetically Breeding Populations of Computer Programs to Solve Problems. Cambridge, MA : MIT Press, 1992.

10. Yeh I.-C. Modeling slump flow of concrete using second-order regressions and artificial neural networks // Cement and Concrete Composites. 2007. Vol. 29. № 6. P. 474–480.

11. URL: http://www.archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test

M. V. Damov
Siberian State Aerospace University named after academician M. F. Reshetnev, Russia, Krasnoyarsk

BACKGROUND RESTORATION IN FRAME AREAS WITH SMALL-SIZE OBJECTS IN VIDEO SEQUENCES

A general concept of removal of artificially overlaid images, natural damages of video images and other small-size objects is presented. The classification of artificially overlaid images is developed. The algorithms of feature points detection and feature points tracking used in video sequence restoration are considered.

Keywords: movement tracking, video sequence, feature points, texture, texture filling.

The task of restoration of video sequences is getting more and more actual as a result of computer engineering development. The restoration of the original images under artificially overlaid object (TV channel logotypes, subtitles etc.) and other small-size objects such as a man, a tree, a stone etc. on a certain background as well as the images distorted as a result of damage of information carrier (scratches on the film etc.) is of primary importance. The solution of this problem in general will result in reduction of costs of video reutilization such as old film remastering, original video forwarding by different TV channels with removal of earlier overlaid, but irrelevant now, computer graphics, and accidentally shot objects, for example, advertising structures.

Overlaid computer graphic images occurred in video can be divided into several groups. There are TV channels logotypes that can be defined as small-size images arranged in one or several frame corners or frame borders; the second group is titles, that is, text areas with information about film makers arranged in any place of a frame; the third group is subtitles, which can be defined as text areas near the top or bottom frame borders with