captionCompression results 3 bit per pixel

Trainig algorithm	ADAM	FTML	Genetic	annealing
MSE	272	254	322	262
PSNR	23.9	24.2	23.1	24.0
PSNR_HVS	24.1	24.4	23.3	24.1
SSIM	0.746	0.756	0.698	0.733
Training time, h	10	10	30	30

captionCompression results 1.5 bit per pixel

Trainig algorithm	ADAM	FTML	Genetic	annealing
MSE	433	397	452	390
PSNR	21.9	22.3	21.7	22.3
PSNR_HVS	22.1	22.5	21.9	22.5
SSIM	0.663	0.673	0.638	0.669
Training time, h	4	4	18	22

captionCompression results 0.75 bit per pixel

Trainig algorithm	ADAM	FTML	Genetic	annealing
MSE	272	254	322	262
PSNR	23.9	24.2	23.1	24.0
PSNR_HVS	24.1	24.4	23.3	24.1
SSIM	0.746	0.756	0.698	0.733
Training time, h	6	6	21	25

that at a separate iteration the parameters of the network being trained don't change significantly, which doesn't correspond to the problem being solved. The CD-k algorithm requires k times more calculations per iteration than CD-1 and at the same time achieves better quality [12], however, in the problem being solved, the values of the gradients are very large and the use of the CD-k algorithm is not advisable.

To compare the effectiveness of the developed learning algorithms in the framework with existing analogues, the strongest optimization algorithms implemented in analogues were taken: the adaptive moment method [13] (ADAM) due to its stability used for training neural networks of complex architecture [14] and following the moving leader method [15] (FTML), and from the current framework — the original annealing method and genetic algorithm.

For training, the first 8000 images were used as a training set, the next 7000 images for the validation set, the remaining 85000 images formed a test set.

To evaluate the compression efficiency, the most common quality functionals were chosen: MSE (mean squared error), PSNR (peak signal to noise ratio), PSNR- HVS (PSNR human visual system), SSIM (structure similarity image measurement).

The experiments were held on the operating system Lubuntu 20.04 with 4-core CPU intel i7-4770k, 16 GB 1600 MHz RAM and GPU nvidia rtx 3070 with 5888 cores. Compiler version gcc 9.4.0, GPU driver version 470.161.03. The framework has been configured to use the GPU for training.

Compilation options "gcc -xc++ -Wl,-z,stack-size=10000000000 superOpenCLFramework.cpp constants.cpp trainPause.cpp deviceConvLayer.cpp devicePoolingLayer.cpp deviceDeepNN.cpp trainDCNN.cpp poolingLayer.cpp convLayer.cpp mlp.cpp autoencoder.cpp testCompression.cpp deviceMLP.cpp RBMGaussBernoulli.cpp deviceRBMGauss-Bernoulli.cpp deepNN.cpp trainDeepNN.cpp deepNNFunctioning.cpp dataProcessing.cpp deviceData.cpp superFrameworkInit.cpp finalTrainStats.cpp finetuning.cpp trainMLP.cpp trainRBM.cpp fastMath.cpp trainSettings.cpp -lstdc++ -$D_FORCE_INLINES -O3 -lOpenCL -lgomp -lm -fopenmp$"

The experiments results are displayed by data compres- sion ratio (see Table 1, 2, 3).

From the experimental results, it can be noted that at low degrees of compression, the annealing method is approximately equal to the gradient methods in terms of the quality of training, but it lags behind them in terms of speed by about 4 times. However, as the compression task becomes more complex, the annealing method begins to surpass them in quality. With 32-fold compression, both random search algorithms significantly outperformed the gradient methods in quality, and the annealing method significantly outperformed the genetic algorithm in the resulting solution quality. The higher the complexity of the problem being solved, the worse the solution is obtained by gradients compared to random search algorithms. Modern training neural networks frameworks either do not support this type of training algorithms in principle, or their functionality in this part is extremely poor, which makes them not the most effective means of solving complex applied problems.

IX. CONCLUSION

The paper presents a software package for training neural networks using random search algorithms. High performance and cross-platform is achieved through the use of OpenMP, OpenCL libraries. A wide supported architectures library in the framework allows us to construct deep neural networks of various architectures, which makes it flexible in applied problems solving. The framework supports a wide variety of computing devices and is able to adapt to low-power computing devices, which makes it possible to use it on a wide variety of devices. Thanks to the random search algorithms support, the framework is able to solve complex applied problems more efficiently than those existing on gradient methods. Using the example of solving the color image compression problem, it was shown that the proposed framework solves neural networks complex training problem more efficiently than existing analogues. From this we can conclude that the developed software package can be