Computer Vision and Pattern Recognition Project: CNN Classifier

Michel El Saliby[1], Federico Pellizzaro[2], and Giovanni Zanin[3]

[1, 2, 3]Master in Computer and Electronics Engineering, University of Trieste

Fall 2023 Course

Contents

1 Introduction

This report presents a project that implements an image classifier based on convolutional neural networks (CNNs). The aim is to compare the performance of several machine learning approaches. All classifiers were trained on a dataset provided by Lazebnik et al. (2006) [5], which includes 15 categories of images.

The evaluation process begins by establishing a baseline through training a CNN in a straightforward configuration. Subsequently, various approaches are explored, both working on the train set and the classifier model. These include the implementation of diverse data augmentation techniques, changing the layout of the CNN, deploying a pre-trained CNN, and a multi-class linear support vector machine (SVM).

2 Problem Statement

The goal is to train an image classifier. The dataset used contains 15 categories of images, namely: office, kitchen, living room, bedroom, shop, industry, tall building, city, street, motorway, coast, open country, mountain, forest and suburb. The dataset was already split into training and test sets, respectively containing 1500 and 3000 images, evenly divided across al classes.

3 Assessment

The assessment procedure is designed to measure the performance, i.e. the accuracy of a given approach on the test set, five times. The resulting score is the sample average of these measurements.

The choice of the number of repetitions has two main constraints. Firstly, stochasticity is introduced in several ways in the learning process, making the performances non-deterministic. Therefore, it is necessary to consider a greater number of realizations to obtain a reliable assessment of a strategy. On the other side, computational costs limit the number of realizations that can be performed.

4 Programming Environment and Libraries

The implementation of this project was conducted using Python, leveraging the Google Colab environment for its computational resources and collaborative features. In addition, the following key Python modules were used:

- `PyTorch v2.1.0`: Used for building and training neural networks.

- `torchvision v0.16.0`: Used for pre-trained models and various computer vision-related functionalities;
- `scikit-learn v1.2.2`: Used for machine learning tasks, including SVM.

For further details about the code and the data, please check Section 8.

5 Implementations

This section discusses the design characteristics of the trained models. Please refer to Section 6 for the results.

5.1 Baseline

In the first phase, a baseline is established by training a CNN according to the specified configurations shown in Figure 1. To fit the network, all input images were resized to 64x64 pixels using anisotropic rescaling.

#	type	size
1	Image Input	64×64×1 images
2	Convolution	8 3×3 convolutions with stride 1
3	ReLU	
4	Max Pooling	2×2 max pooling with stride 2
5	Convolution	16 3×3 convolutions with stride 1
6	ReLU	
7	Max Pooling	2×2 max pooling with stride 2
8	Convolution	32 3×3 convolutions with stride 1
9	ReLU	
10	Fully Connected	15
11	Softmax	softmax
12	Classification Output	crossentropyex

Figure 1: Layout of CNN employed as baseline (CNN_1)

The training set was split into 85% for actual training and 15% for validation. *Stochastic gradient descent* (SGD) with momentum optimization algorithm was employed, with mini-batches of size 32. The initial bias values were set to 0, and the initial weights were drawn from a Gaussian distribution with a mean of 0 and a standard deviation of 0.01. Cross entropy was used as loss function.

The employed stopping criterion is based on the loss on the validation set. Two hyper-parameters were introduced: `max_epoch`, to define the maximum number of epochs that can be performed, and `max_patience`, which defines how many epochs to wait before stopping the training if the minimum validation loss does not improve. In the following experiments each training session will be performed with `max_epoch`=60 and `max_patience`=12.

The *learning rate* and the *momentum* were fine-tuned to optimize performance. Initially, two discrete sets of potential values were defined, one for each parameter. The learning rate was selected from the set {0.0001, 0.001, 0.01, 0.1, 0.5}, while for the momentum the set {0.1, 0.3, 0.5, 0.7, 0.9} was chosen. The fine-tuning process involved evaluating each possible combination from these sets, as outlined in Section 3. From now on, the network described in this section will be referred to as CNN_1.

5.2 Initial improvements

The first attempt to improve the performances of the initial network was to train a new model (CNN_2) that introduced:

- batch normalization immediately before the ReLUs (Ioffe et al. (2015)[4]);
- dropout with probability 25% (according to results found by Srivastava et al. (2014)[6]);
- gradual increase along layers of the kernel size in the convolution layers (3x3, 5x5, 7x7);
- combination of the loss function with an L2 regularization factor (with weight decay 0.008);
- standard deviation of the initial weights specified as a parameter.

Furthermore, the new model was tested with an augmented train set obtained by:

- extracting from every image a random crop large at least 70% of the original, rotating it by a random angle in $[-10°, 10°]$ and adding the resultants to the train set;
- adding to the images above all the respective flipped ones, by rotating on the vertical axis.

Afterwards, the CNN (and the augmented train set) described above where employed in an ensemble of networks (EoN) as described by Szegedy et al. (2015)[2], which consisted in the training of a certain number of networks (5 by default) and the computation of the output category as the maximum between the means of the networks' outputs.

Finally a variant of the first, i.e. CNN_2, attempt was created (CNN_3), with the addition of a convolution (9x9 kernel size) and a fully connected layer before the output one.

The stopping criterion adopted for the strategies in this section was the same adopted in the previous one. The portions of the train-validation split were initially (i.e. before the train set augmentation) the same (85% - 15%).

The best-performing values of the learning rate and the momentum, obtained through grid search on the baseline, were applied. However, in Section 6 are presented only the results obtained using *Adam* as the optimizer, in this case default initial and adaptive momentum values were used.

5.3 Pre-trained networks

Pre-trained AlexNet (Krizhevsky et al. (2012)[1]) was used for this step. The network was loaded and its last fully connected layer was replaced with one having the right number of outputs, i.e. 15 as the number of image classes. The weights of all layers were frozen except for the weights of the last fully connected one. All images in the dataset were adjusted to meet AlexNet's specifications, including resizing and normalization. Standard deviation and mean values were used consistently with those established and applied in the ImageNet dataset during the normalization process, as outed by Vezakis et al. (2023)[7].

Employing both Stochastic Gradient Descent (SGD) and Adam optimizers, AlexNet underwent testing with 4 variations: AlexNet_1 and AlexNet_2, utilizing the original dataset with SGD and Adam optimizers respectively, and AlexNet_3 and AlexNet_4, utilizing the augmented dataset (described in Section 5.2) with SGD and Adam optimizers respectively. When using SGD, the learning rate and the momentum chosen were the ones that had the highest score after the fine-tuning process described in Subsection 5.1. On the other hand, when using Adam, default initial and adaptive momentum values were used.

5.4 Feature extraction for SVM

The pretrained AlexNet was employed again, this time serving as a feature extractor for an SVM. For every image in the training set, its output at the last AlexNet's hidden layer was computed, resulting in 256 6x6 features for each image. Then, the features were flattened and used with the train set labels to train SVM with different kernels, i.e. *linear*, *(non-homogeous) polynomial*, *radial basis function (gaussian)*, *sigmoid*; and different multi-class classifiers, i.e. *One-vs-Rest* (OvR), *One-vs-One* (OvO), *Error Correcting Output Code* (ECOC) (Dietterichet al. (2023)[3]). The obtained classifiers were then assessed on the test set. Due to computational constraints, data augmentation was not explored in this step. Furthermore, the assessment procedure outlined in Section 3 was not executed for the models discussed in this section, as, for a fixed dataset, SVM exhibits deterministic behavior. Stochasticity might be introduced by sampling a random subset of the training set for each training session. This approach was not pursued due to computational limitations. Therefore, for each model only one training session with the whole dataset was executed.

6 Results

Starting from the baseline network described in Section 5.1, the fine-tuning results are presented in Table 1. The highest score was achieved by the pair

Momentum	**Learning Rate**				
	0.0001	0.001	0.01	0.1	0.5
0.1	0.074	0.176	0.168	0.067	0.052
0.3	0.084	0.216	0.152	0.063	0.063
0.5	0.070	0.254	0.146	0.066	0.067
0.7	0.076	0.254	0.108	0.056	0.074
0.9	0.173	0.267	0.072	0.067	0.075

Table 1: Scores of each hyper-parameters pairs tested in Section 5.1. Welch's t-tests were conducted to evaluate the null hypotheses $H_0 : \mu_{\text{best pair}} = \mu_{\text{other pair}}$. The cells are color-coded: the maximum score is highlighted in green, orange cells represent p-values greater than or equal to the threshold 0.1, and white cells represent p-values smaller than the threshold.

(0.001, 0.9) -where the first component is the learning rate and the second is the momentum- with an average accuracy of 26.7%. A Welch's t-test was conducted between the best-performing pair and each of the other pairs to assess the null hypothesis $H_0 : \mu_{\text{best pair}} = \mu_{\text{other pair}}$. The Welch's t-test was chosen because the assumption of equal variance across all pairs is not met. The threshold for the p-value was set to 0.1 instead of 0.05, as is frequently done in the literature to address the limited number of samples measured for each pair. As shown in Table 1, the null hypothesis cannot be rejected for pairs (0.001, 0.5) and (0.001, 0.7). Therefore, while it is evident that the optimal learning rate is 0.001, the specific value of the momentum appears to be less crucial, as long as it is higher than 0.5. For brevity, the pair (0.001, , 0.9) will be chosen as the reference for the baseline results in the remainder of the discussion.

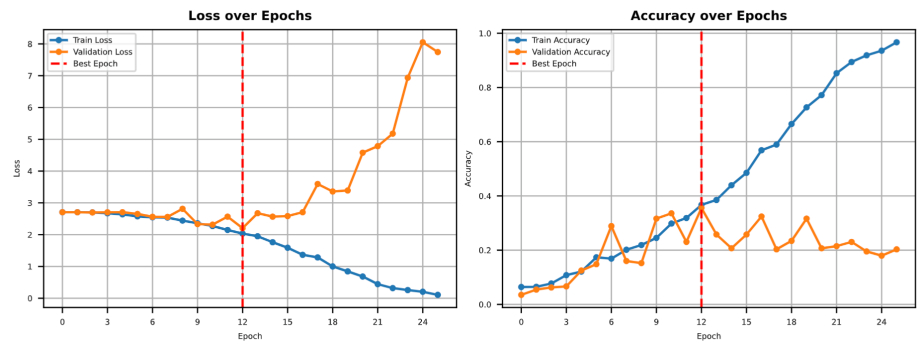

Figure 2: CNN_1 with lr = 0.001 and mt = 0.9: loss and accuracy convergence on training and validation set during training. The dashed red line indicates the epoch at which the model achieved the lowest loss on the validation set.

A final training of CNN_1 with the aforementioned parameters was conducted to generate the convergence plot in Figure 2 and the confusion matrix in Figure 3. This particular training session achieved an accuracy on the test set of 25%. As depicted in the confusion matrix, CNN_1 made trivial mistakes, such as confusing bedroom with industrial images.

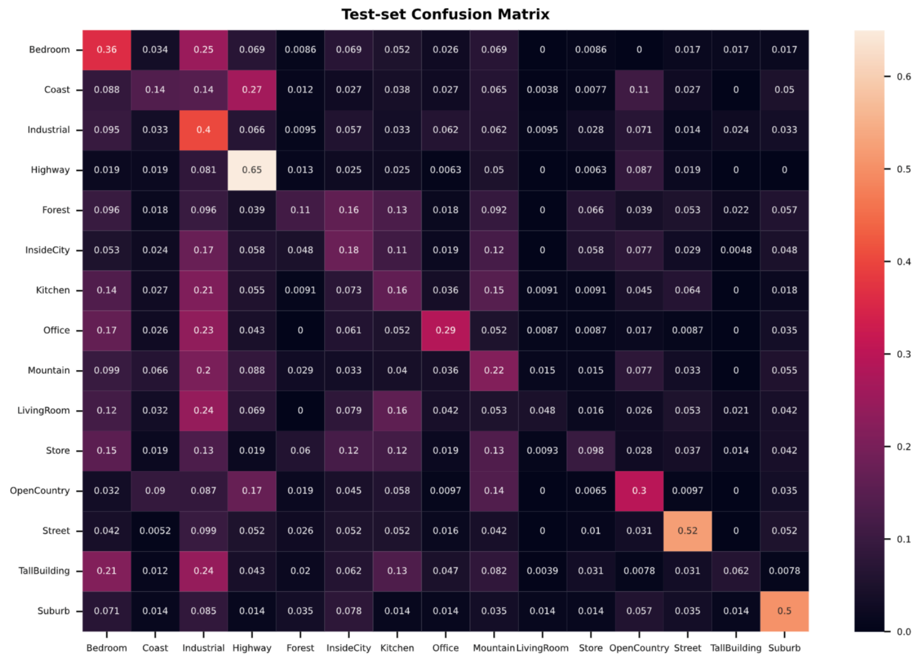

Figure 3: CNN_1 with lr = 0.001 and mt = 0.9: confusion matrix for the test set.

Scores obtained by all networks described in Section 5.2 are higher than the baseline, with each model achieving at least an average accuracy of 60%. Results are presented in Table 2. The ensemble of networks (EoN) emerges as the most effective among the tested models, not only achieving an average test accuracy of 66.4% but also exhibiting a smaller variance among realizations. Furthermore, conducting Welch's t-test between the scores obtained by EoN and the other networks, namely CNN_2 and CNN_3, reveals that all p-values are smaller than the threshold of 0.1.

Having achieved the highest score among the models tested in Section 5.2, a single EoN was trained once more to plot its confusion matrix on the test set, as shown in Figure 4. The confusion matrix, associated with an accuracy of 67%, reveals that the EoN not only performs more than two times better than the baseline but also makes more reasonable mistakes, such as confusing bedroom with the living room or open country with the coast.

Concerning the pre-trained models described in Section 5.3, as illustrated in Table 3, the highest score was achieved by AlexNet_1, i.e. pre-trained AlexNet

Network	**Average accuracy**	**Accuracy Variance**
EoN	0.664	0.000037
CNN_2	0.617	0.000646
CNN_3	0.616	0.001451

Table 2: Scores of each network tested in Section 5.2. Welch's t-tests were conducted to evaluate the null hypotheses $H_0 : \mu_{\text{best network}} = \mu_{\text{other network}}$. The cells are color-coded: the maximum score is highlighted in green, white cells represent p-values smaller than the threshold 0.1.

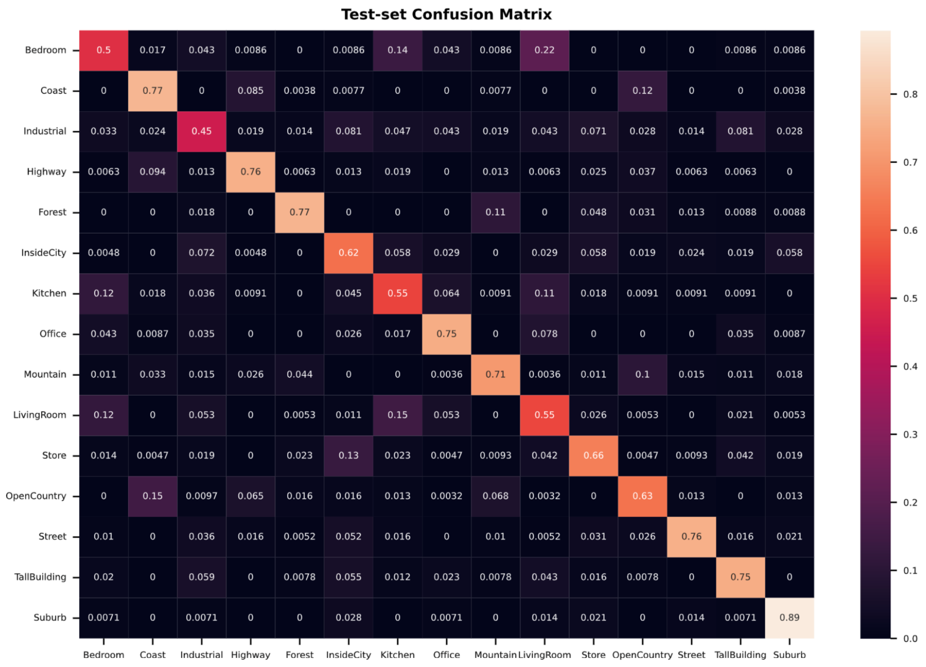

Figure 4: EoF, 5 ensemble of CNN_2 networks, confusion matrix for the test set.

with stochastic gradient descent and without data augmentation. The results suggest that employing a pre-trained CNN makes the subsequent training less dependent on the dataset and optimizer. Not only all scores are similar, but also when conducting Welch's t-test between the scores obtained by AlexNet_1 and the other networks, only AlexNet_4 had a p-value smaller than the threshold of 0.1, even though it was equal to 0.085.

The results discussed above provided no conclusive evidence for necessarily choosing AlexNet_1. Therefore, AlexNet_2 was selected as the reference for Section 5.3 because it was the least time-consuming during training. Figure 5 displays the confusion matrix on the test set for an additional training session of AlexNet_2, resulting in a model with an accuracy of 86%. The confusion matrix reveals that the most common misclassifications made by AlexNet_2 are

Network	Average accuracy	Accuracy Variance
AlexNet_1	0.862	0.000007
AlexNet_2	0.860	0.000018
AlexNet_4	0.859	0.000025
AlexNet_3	0.854	0.000060

Table 3: Scores of each network tested in Section 5.3. Welch's t-tests were conducted to evaluate the null hypotheses $H_0 : \mu_{\text{best network}} = \mu_{\text{other network}}$. The cells are color-coded: the maximum score is highlighted in green,orange cells represent p-values greater than or equal to the threshold 0.1, and white cells represent p-values smaller than the threshold..

similar to those made by EoN. Notably, AlexNet_2 scores approximately 1.3 times better than EoN and more than 3 times better than the baseline.

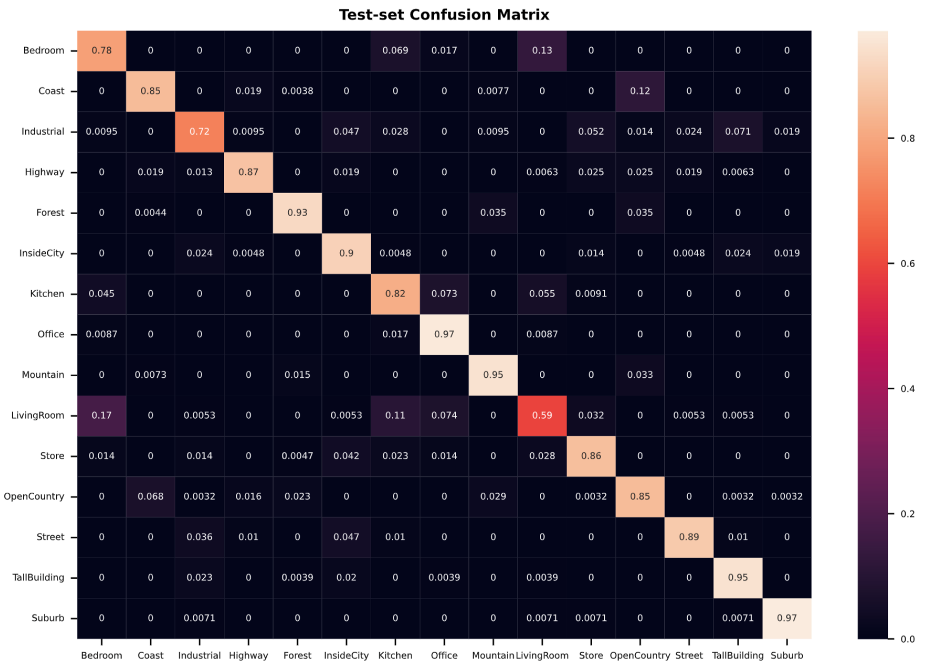

Figure 5: AlexNet_2, confusion matrix for the test set.

Finally, when employing AlexNet as a features extractor for a multi-class SVM as described in Section 5.4, the models employing One-vs-One multi-classifier were the best ones, both for sigmoid and linear kernels.

The highest accuracy of 87% achieved by the models in Section 5.4 is only marginally larger than the 86% accuracy achieved by AlexNet_2. Furthermore, by comparing the confusion matrices in Figure 5 and Figure 6, it becomes evident that the mistakes made are similar, with a notable instance being the confusion between bedroom and living room. This suggests that despite the dif-

Classifier	**Kernel**			
	Linear	**Polynomial**	**Radial Basis Func.**	**Sigmoid**
OvR	0.851	0.841	0.857	0.848
OvO	0.871	0.808	0.866	0.872
ECOC	0.791	0.793	0.855	0.764

Table 4: Accuracy on the training set for different SVM kernels and multi-class classifiers.

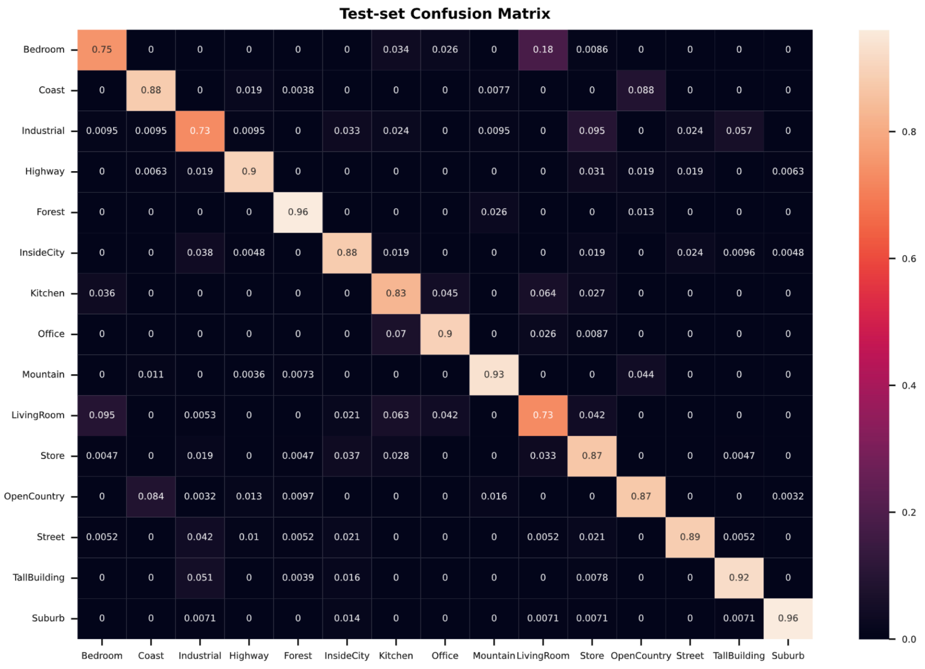

Figure 6: SVM with OvO multi-classifier and sigmoid kernel, confusion matrix for the test set.

ference in model architectures, there are shared challenges in classification, for example Figure 7 shows an image of the test set that is labeled as ”bedroom”: it is reasonable to assume that also a human-classifier could probably classify this as a living room.

7 Conclusion

In conclusion, the comprehensive evaluation of various models in Section 5 highlights pre-trained AlexNet, particularly as represented by AlexNet_2 and as a feature extractor for an OvO SVM, as the most robust and effective choice. This preference is grounded in both superior accuracy and ability to deal with a

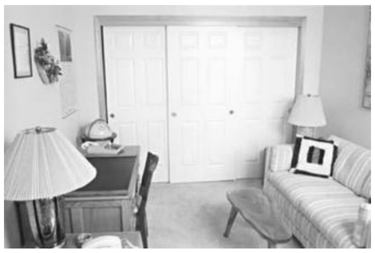

Figure 7: An image of the test set labeled as ”bedroom”.

fuzzy classification challenge, as evidenced by the reasonable misclassifications observed in the confusion matrices.

8 Reproducibility

Code, data, and datasets are available in the folder accessible through the following link: `https://shorturl.at/bdguN`.

To execute the notebooks correctly, you must adjust the `filepath` at the beginning of each notebook to match the path of your Google Drive directory tree.

The folder contains four Google Colaboratory notebooks, each corresponding to a subsection of Section 5. Additionally, two utility modules with the extension `.py` are included, containing frequently used functions. Experiment results are available in files with the extension `.csv`. The code is thoroughly commented, providing a guide for understanding the meaning of code and data files. Lastly, two folders containing the training set and test set are provided.

References

[1] Krizhevsky Alex, Sutskever Ilya, and Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks. 25:1097–1105, 2012.

[2] Szegedy Christian, Wei Liu, Yangqing Jia, Sermanet Pierre, Reed Scott, Anguelov Dragomir, Erhan Dumitru, Vanhoucke Vincent, and Rabinovich Andrew. Going deeper with convolutions. pages 1–9, 2015.

[3] Dietterich Thomas G. and Bakiri Ghulum. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, 2(1):263–286, jan 1995.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[5] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. 2:2169–2178, 2006.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[7] Vezakis, Ioannis A., George I. Lambrou, and George K. Matsopoulos. Deep learning approaches to osteosarcoma diagnosis and classification: A comparative methodological approach. *Cancers*, 15(8):2290, 2023.