Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем

Classification of buildings images by damage level

Текстова частина до курсової роботи за спеціальністю «Комп’ютерні Науки»

Керівник курсової роботи
ст. викладач
каф. мультимедійних систем
Кузьменко Д. О.
_____(підпис)
“__ ”_____ 2025 р.

Виконала студентка 3 р. н.
Кузнецова А. В.
“ __”_____ 2025 р

Календарний план виконання курсової роботи

№	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Визначення теми кваліфікаційної роботи	Жовтень 2024 р.	
2.	Опрацювання літератури	Листопад-Грудень 2024 р.	
3.	Виконання практичної частини	Грудень-Березень 2025 р.	
4.	Написання текстової частини курсової	Квітень 2025 р.	
5.	Оформлення презентації для захисту	Травень 2025 р.	
6.	Захист курсової роботи	15 травня 2025 р.	

Студентка: Кузнецова А. В.

Керівник: Кузьменко Д. О.

Table of Contents

1 ABSTRACT

This course paper focuses on the development of a machine learning model for classifying destroyed buildings based on the type of damage caused. It explores and compares different approaches to model creation and training, using real-life data of buildings in Ukraine that were damaged as a result of Russian military aggression. The study aims to contribute to effective damage estimation and planning for urban reconstruction in the context of the ongoing war in Ukraine.

Key words: Image Classification, Shelling, Convolutional Neural Network, Deep Learning, Computer Vision

2 INTRODUCTION

The shelling in Ukraine is causing massive destruction of infrastructure facilities on a daily basis [1], and a quick and accurate classification of the extent of damage may be crucial for recovery planning, distribution of humanitarian aid, and recording of war crimes. Automating the classification of damage types using machine learning can significantly speed up and standardise this process, and enable quick decision-making for timely relief.

The goal of the work is to investigate the effectiveness of different neural network architectures [2] and approaches to image classification [3] in the specific context of identifying the nature of building damage. In particular, the development and implementation of a CNN [4] with an Image Fusion mechanism [5] for event classification makes a significant contribution to the development of computer vision methods [6] for analysing the consequences of armed conflicts. Our approach overcomes the limitations of standard image classification methods by aggregating information from several images of the same scene, which allows for greater classification accuracy.

The aim of the study is to develop and evaluate the effectiveness of machine learning models for automated classification of building damage by type, using real data of destroyed objects in Ukraine as a result of Russian aggression. We have outlined the following steps of this work:

1. Finding a relevant dataset with real-world data and its further analysis.
2. Conceptualizing theoretical concepts of image classification for our task.
3. Creating and training a classification model based on the ResNet50V2 architecture.
4. Research and implementation of the Image Fusion method in CNN architecture

5. Evaluation of the effectiveness and comparison of the results of different approaches.

The object of research is the process of classifying building damage by its type, using deep learning [7] and computer vision methods.

The subject of the study is machine learning methods and models for classifying images of destroyed buildings, in particular, the ResNet50V2 architecture [8] and CNN with the Image Fusion mechanism, their efficiency and accuracy in determining the type of damage caused.

The practical significance of the results obtained lies in the possibility of their application for:

- Estimating the scale of destruction - the developed classification system makes it possible to systematise and quantify infrastructure damage by type of attack.
- Optimisation of the reconstruction process - automated classification allows faster decision-making on the necessary methods to restore damaged facilities.
- Documentation of war crimes - the model can serve as a tool for objectively recording the types of weapons used against civilian objects.

In this work, we have used: Google Colab [9] with free GPUs [10] as development environment; Python 3.11 as a main programming language; Torchvision [11] - as main computer vision framework for image transformations; TensorFlow/Keras and ResNet50V2 [12] - as the first implementation of the model and PyTorch [13] as the main framework for the second implementation of the model.

This coursework is divided into sections, each of which covers important aspects of developing and comparing machine learning models for a given task.

Section 3 provides the theoretical background, including an overview of the ResNet architecture, the principles of Image Fusion, and methods for evaluating classification models.

Section 4 describes practical implementation and experiments, including data preparation and implementation of both approaches. Section 5 presents the experimental results.

Section 6 highlights the conclusions of the study with an analysis of the results.

3 THEORETICAL FRAMEWORKS

3.1 Machine Learning and Deep Learning Approaches for Image Classification

With the development of machine learning (ML), approaches to image classification are constantly innovating. Traditionally, ML image processing methods need to have domain knowledge and rely on manually created features to identify relevant patterns in images [14]. These approaches are known to be effective for simple tasks, but they still often fail to cope with the complexity and variation inherent in real-world images, especially in complex tasks such as damage assessment.

With further advancement of technology, a new and more powerful method of image processing has become widely used - Deep Learning (DL) [15]. In a big difference from traditional methods, DL models can extract features from images directly from the raw pixels and automatically learn hierarchical representations from data. These qualities make Deep Learning very suitable and flexible for more complex computer vision tasks, where specific image patterns could be very hard to extract.

This work focuses on real-life data of damaged buildings, so we chose the Deep Learning approach, as it provides significant benefits through its ability to identify complex textures, contextual information, and subtle structural patterns that characterise different types of damage.

Deep learning is based on neural networks (NNs) [16]. NNs are computational models that function similarly to neurons in the human brain. A neural network consists of layers of artificial neurons that are interconnected and process input

data, transforming it into the result of creating classifications, in our case. The basic architecture of NNs includes:

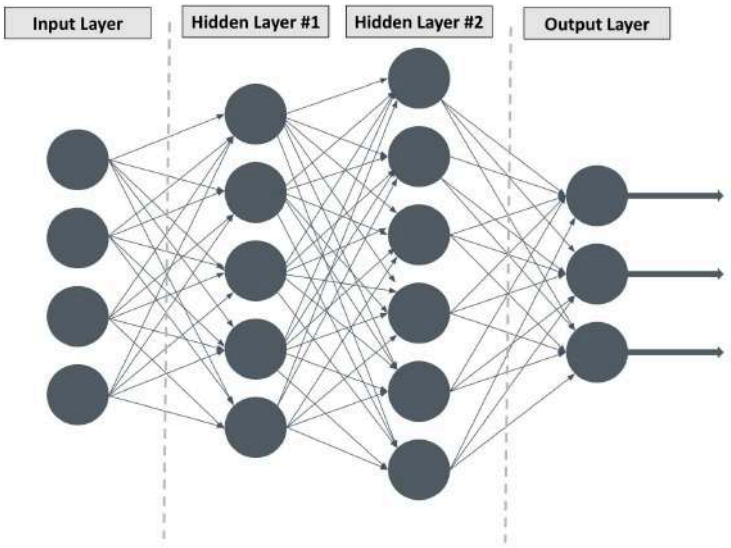

Figure 1. Example of the structure of a fully-connected neural network.

As shown in Figure 1, the basic architecture of NNs includes:

- An input layer that receives raw data (in our case, image pixels).
- Hidden layers where information is being processed.
- An output layer that produces the final prediction (in our case, damage classification classes).

The layers in neural networks can be inconsistent. Each neuron applies a mathematical function to its inputs, which is often followed by a non-linear activation function that adds complexity to the model [16]. The network learns by adjusting the weights of the connections in the hidden layer within backpropagation, which reduces the difference between the predicted results (NN decision) and the actual values (labels). The number of hidden layers can significantly affect its ability to learn. It is also considered that the more hidden layers a network has between the input and output layer, the deeper it is. Deeper

networks can capture more complex and abstract features, which is crucial for distinguishing subtle differences in the pattern of damage caused by different types of attacks. However, the deeper networks are - the harder it becomes to train them.

3.2 Introduction to Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [17] are specialized neural network architectures that are designed for processing grid-like data, such as images. CNNs can be used for diverse types of computer vision tasks, but in our work, we use them to classify images for damage from various types of shelling. In contrast to standard neural networks, CNN has its specific architecture of hidden layers that makes it different. Every CNN consists of such layers in its structure:

- Convolutional layers
- ReLU layers
- Pooling layers
- Fully connected layer (the final layer)

There is no fixed number of layers, and each can occur several times. The order of them can also be inconsistent, but it is important to note that there are some strict rules: the CNN must start with a convolutional layer and end with a fully connected layer [18]. And there must also be a final pooling layer before the last fully connected one.

An image classifier takes the numerical pixel values of an image, passes them through the CNN, and gets the final output [18]. In our work, the output would be the class with the identified type of shelling: Ground attacks (shelling, mines, artillery, ammunition) and Air attacks (drones, missiles, and rockets).

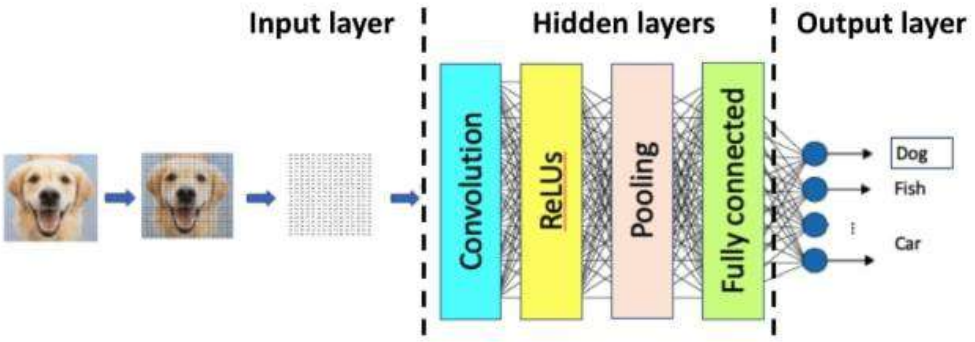

Figure 2. Simple CNN classifier example

Figure 2 shows a simple example of how an image is being processed within the hidden layers of a CNN and then being classified as one class from multiple possible options.

Convolution layer

Convolution layers are the basic building blocks in image classifiers. Convolution looks for and extracts image features, a specific characteristic of the input image, such as the edges or shape of a particular object [18]. The feature is then converted into a matrix of numerical pixel values. This matrix is then used as a feature detector.

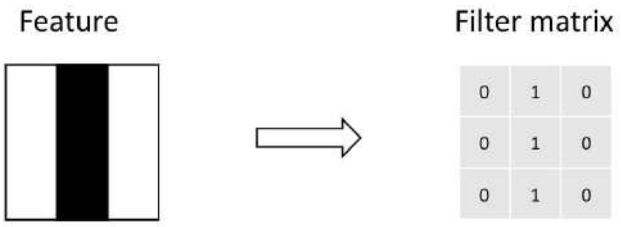

Figure 3. Vertical line detector example

In Figure 3, it can be seen how the feature is being extracted from the vertical line to a matrix with pixel values on the vertical line.

To scan an image, the filter matrix, or *kernel,* is moved across the image, pixel by pixel. A value is calculated for each overlay, which depends on how well the filter matches the image. The calculation is a multiplication of two matrices. Thus, the better the match, the higher the resulting value in the feature map.

Figure 4. Image transformation to a feature map by multiplication of matrices

Figure 4 describes how an image is transformed into a feature map.

Convolution: Image matrix * Filter matrix = Feature map.

Rectified Linear Units Layer

Rectified linear units (ReLU) are a common choice for an activation function [18]. The activation function is necessary for adding non-linearity to the computation. ReLU takes the feature map (it could be the output from the previous convolution layer) and rectifies every negative value to zero, without changing the positive numbers. Nonlinearity implies that the slope is not constant. The ReLU level is nonlinear because:

- for x < 0: the slope of the function is 0.
- for x > 0: the slope of the function is 1.

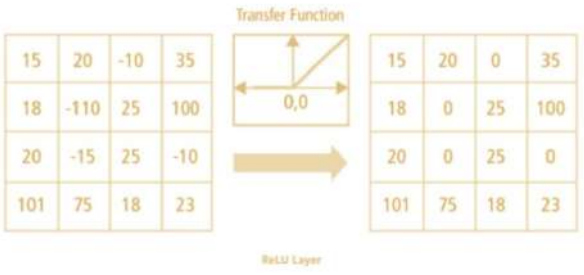

Figure 5. Rectification of all negative values to zero by the ReLU layer.

Figure 5 shows an example of how the ReLU layer rectifies the negative values of the matrix (-10, -110, -15, -10) to zero.

Pooling Layer

Within the process of the pooling layer, the input matrix is being processed: a (2x2) filter runs over the feature map and assigns a single value per subregion to the new output matrix [18]. The main goal of this process is to downsize an image so that the computational speed of the image classifier is accelerated. Max pooling [19] is considered the most common pooling approach.

In the max pooling approach, the filter, after running over the input matrix, takes the maximum value of a specific position and assigns it as a value to the output matrix.

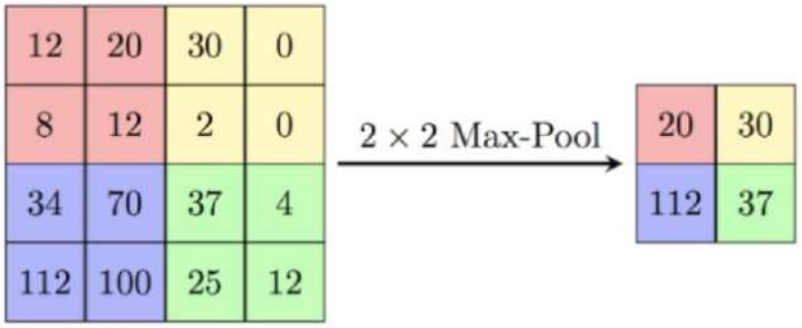

Figure 6. Max-Pool example

An example in Figure 6 illustrates how a 2x2 filter slides over a 4x4 matrix, selecting the maximum value from each subregion and assigning it as a single value in the output matrix. To be specific, in the four fields of the bottom right, the highest value is 37. Then the value is assigned as a single field to the output matrix. And the filter goes on next to every position. As a result, the output matrix has each value corresponding to the maximum value of the related subregion.

Fully connected layer

The CNN is completed with a fully-connected layer [18]. It takes the output from the last pooling layer as input and then aggregates all the information, applies it to the weights, and makes the classification at the end. Within the process of the fully connected layer, each neuron computes a weighted sum of inputs, producing logits (raw, unnormalized prediction scores). These logits are then transformed using the softmax function [36], which converts them into a probability distribution by raising each value to an exponent and normalizing so they sum to 1. This produces probability values between 0 and 1 for each class. Finally, the argmax operation selects the class with the highest probability as the final prediction, completing the classification process.

3.3 ResNet architecture

The Residual Network (ResNet) family of models has made a major impact within the Deep Learning revolution in the domain of computer vision [20]. Researchers from Microsoft Research were the first to develop the idea of residual learning, which has led to models learning as depth increased. ResNet was introduced to the world in the paper Deep Residual Learning for Image Recognition in 2015 at Conference on Computer Vision and Pattern Recognition [21].

As Deep Neural Networks have significantly increased in accuracy for various computer vision tasks, they have had several problems with model degradation and vanishing gradients that led to worse performance than some shallower models. The architecture of ResNet overcomes all these drawbacks and enables training with hundreds of layers. By incorporating residual connections, ResNet effectively solves the problem of the vanishing gradient [21]. Furthermore, the residual

connections allow gradients to pass directly through the network, which results in easier deep model training and even higher result achievement.

In this work, we use ResNet50V2 architecture [8], as it is proven to have solid performance and flexibility in various computer vision applications, including custom image recognition tasks. ResNet50V2 is a configuration of ResNet with 50 layers.

The layers of ResNet50V2 architecture are: convolutional, pooling, fully connected layers, and residual blocks that make use of skip connections [21]. For this architecture, it is essential to be based on the residual block concept, which makes the input and output parameters trainable. It contains:

1. Input Layer takes an image as input.
2. Convolutional Layers extracts features from the input.
3. Bottleneck Design: each residual block consists of three layers:
    - 1x1 convolution,
    - 3x3 convolution,
    - 1x1 convolution.
4. Identity and Residual Connections: they overcome the vanishing gradient problem in deep networks and allow gradients to flow freely through the network during backpropagation. Identity Connections bypass paths that directly transmit input data to the output, skipping convolution layers, and Residual Connections allow the network to learn residual mapping F(x) + x, rather than full mapping H(x)).
5. Pooling Layers lead to a reduction in spatial dimensions of the feature maps.
6. Fully Connected Layers process and perform the final classification.
7. Output Layer infers the class probabilities.

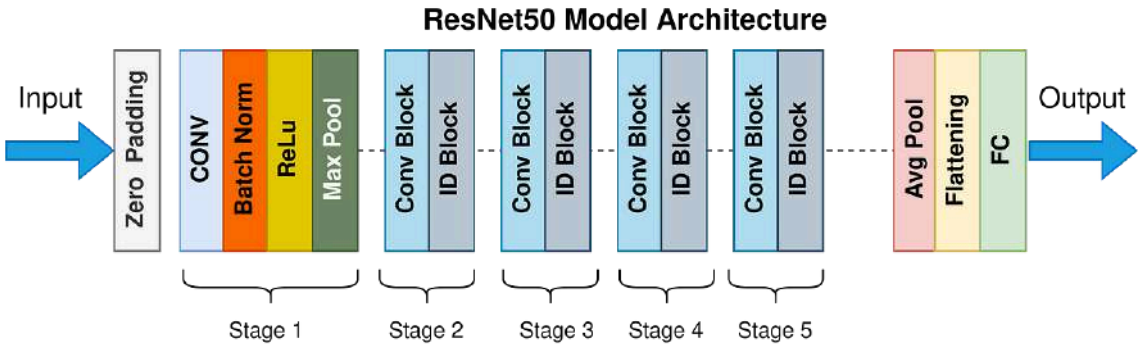

Figure 7. ResNet50 model architecture

Figure 7 shows the architecture of the ResNet50 model. The model processes the input data through 5 sequential stages [21]:

- Stage 1 starts with Zero Padding, followed by a convolutional layer, and then by Batch Normalisation, ReLU activation, and Max Pooling in order to reduce the dimensionality. This stage is similar to traditional CNN logic.
- Stages 2-5 are followed by a series of Conv Blocks and ID Blocks that together form the residual architecture of the network. Every block consists of convolutional layers with residual connections that are meant to prevent the traditional CNN's problem of gradient vanishing.
- Final stage: the convolutional blocks are preceded by Average Pooling for wide averaging of features, then flattening for converting the representation to a one-dimensional vector, and a final fully connected layer for classification, which infers the prediction of the network.

Aside from the layers and architecture of the model, it is important to mention the concept and importance of Batch Normalisation [22]. It is a technique that normalises the output values of each convolutional layer. It calculates the mean and variance for each mini-batch of data, which is then passed to the activation function. Placing Batch Norm after each convolutional layer in the Conv and Identity Blocks provides several important benefits, such

as: accelerating training, regularising and reducing overfitting, and stabilising the activation distribution. This technique is important for the successful training of deep networks.

The Conv Block and Identity Block are the key components of the ResNet50 model that differ from traditional CNNs and make its architecture unique [21].

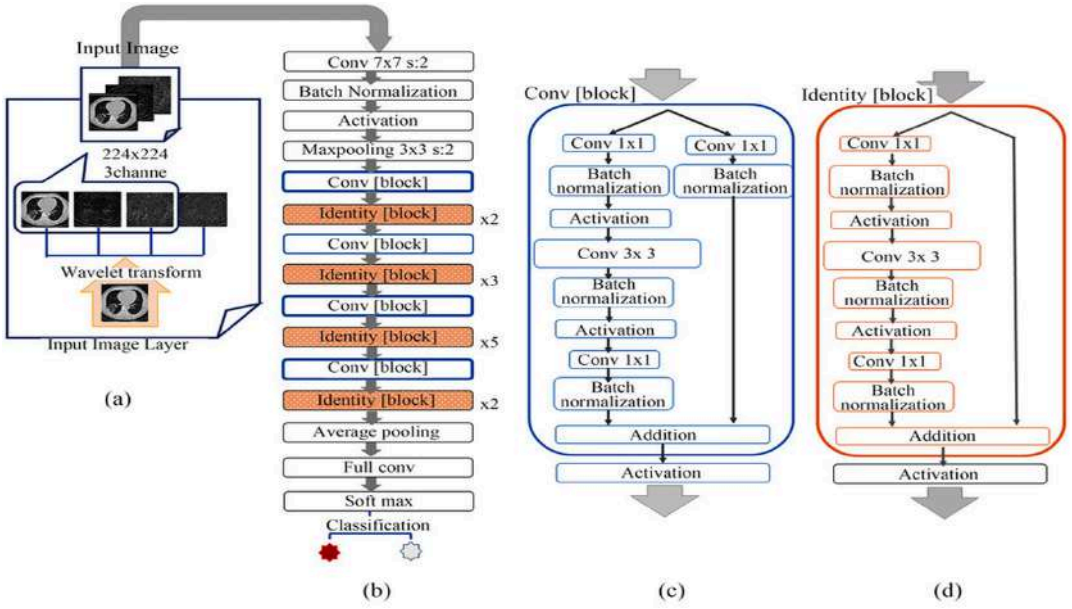

Figure 8. Detailed ResNet50 model architecture

Figure 8 describes a more detailed structure of Conv Block and Identity Block [23]. Section (c) shows in detail the Conv Block, specifically the sequence of three convolutional layers (1x1, 3x3, 1x1), each of which is preceded by a batch normalisation and an activation function. The skip connection is also shown in Figure 8, and implemented through the Addition operation. Section (d) illustrates the Identity Block with a similar structure, but it has a key difference in the connection scheme. Both blocks end with the operation of adding the input and processed signals, which are followed by activation.

Residual blocks combine multiple convolution layers with a skip connection or shortcut [21]. That means that the result at the beginning of each layer is appended to the output of the end-of-layer, leading to a quick transition between layers to ensure that the gradient does not vanish. Instead of learning the mapping H(x), the residual blocks learn the mapping F(x) = H(x) – x, and this makes it easier to optimise the network [24]. Skip connections enable gradients to flow directly through the network, stopping them from becoming too shallow as they propagate in the opposite direction. This approach allows training very deep networks.

There are other different depth ResNet models, such as ResNet152, ResNet101, ResNet34, and ResNet18. We also use ResNet18 [25] in the second approach of the experimental research. The main difference between ResNet18 and ResNet50 [26] is their depth and block structure: ResNet18 has 18 layers and uses simple Basic Blocks, each containing two 3×3 convolutional operations with a direct residual connection. It is a smaller model with fewer parameters (about 11.7 million), whereas ResNet50 has about 25.6 million parameters.

The ResNet50 model has many advantages [21]:

- good performance with high accuracy on large datasets with images, where the patterns are very subtle and hard to extract;
- efficient usage in training deep networks with residual connections;
- transfer learning with pre-trained ResNet50 models, which we also incorporated in our work.

Although the model is very powerful, it still has some limitations, like large computational cost and training complexities due to the big time and memory requirements.

3.4 Image Fusion

Image fusion (IF) is the process of collecting all the relevant information from several images and incorporating it into a single image [27]. This condensed image is more accurate and informative for the model than any other separate image. In addition to reducing the amount of data, IF aims to make the model input better understandable for analysis and processing. There are different IF levels to perform based on the stage of fusion accomplishment.

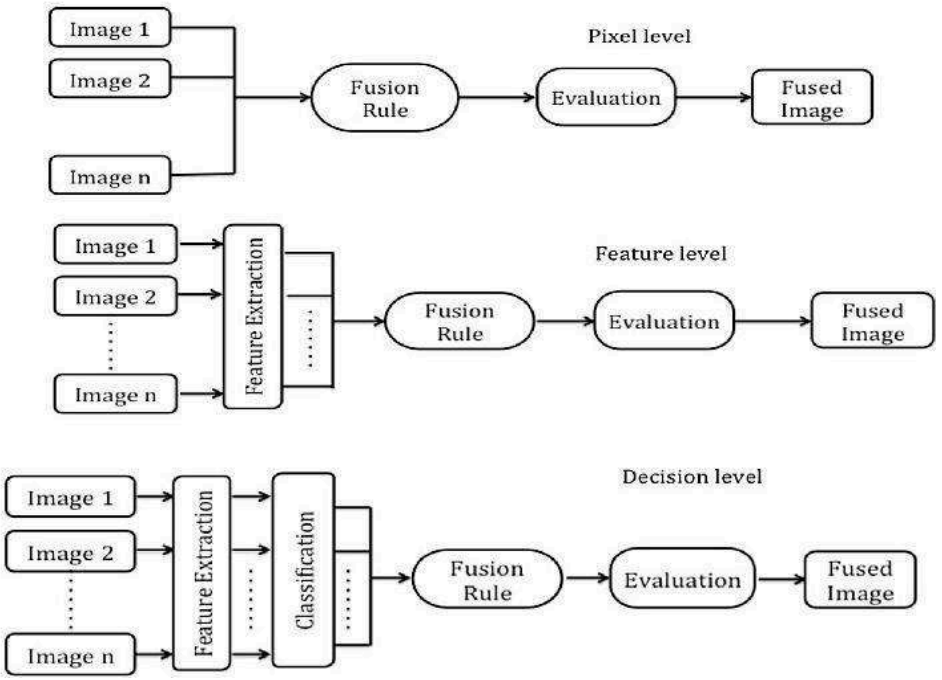

Figure 9. Image Fusion Levels

Figure 9 shows the three levels of Image Fusion [28]:

- Pixel level: This level is considered low because of the simplicity of the method. It is based on extracting features from two input images and creating a single average image.
- Feature level: It validates characteristics (size, color) from multiple images, and creates an enhanced image after extracting the features.
- Decision Level: It is a high-level method because of its multistage representation usage and region-based measurements calculations.

There are also different types of Image Fusion [28]:

- Single-sensor IF: Its algorithm merges a set of images and creates a new image with the best information content. Single-sensor image fusion perceives the real world as a sequence of images. This method has certain disadvantages, such as the limitations of the image sensor that can be used in certain sensing areas. For instance, some sensors are well suited for brightly lit environments, such as sunlight during the day, but may not be good for dark and fuzzy environments, such as night.
- Multi-sensor IF: This method combines images from multiple sensors to form a fused image. For example, a digital camera and another infrared camera capture their images, and the fusion produces the final composite image. This approach avoids the disadvantage of using a single sensor. Taking the previous example, a digital camera would be well suited for highly lit environments, while at the same time, an infrared camera would be good for dark environments.
- Multiview IF: This method uses images that have multiple or different types of views at the same time. This approach uses images from different environments, such as multispectral, clear-visible, and infrared sensing.
- Multi-focus IF: This method is used to process images from 3D views. It works by dividing the original image into areas so that each area is in focus for at least one image channel.

There are many ways to integrate Image Fusion, but the most common would be within CNN [28]. In 2021, for medical application purposes, a CNN-based fusion framework for feature extraction and image retrieval was developed using a sophisticated loss function. To monitor the activity level and feature integration, the authors decided to use the CNN as part of the general fusion structure.

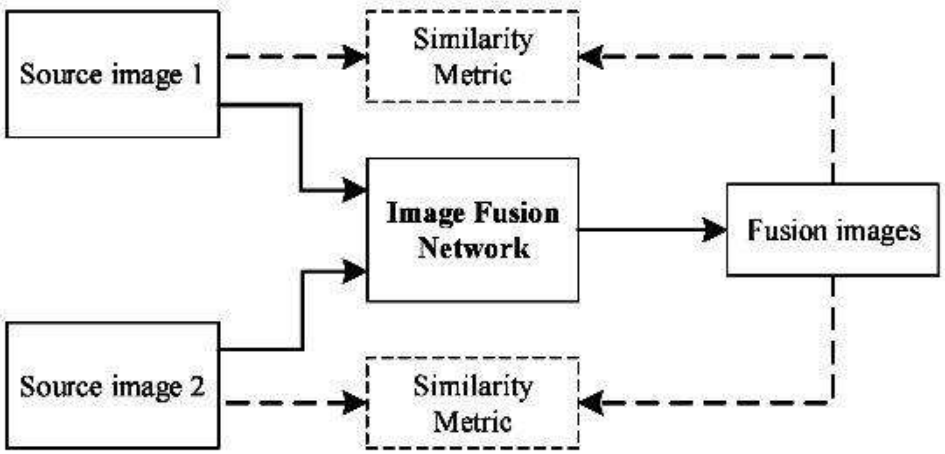

Figure 10. IF implementation within CNN

Figure 10 shows the scheme of Image Fusion implementation using CNN [28]. In this case, to perform medical IF, the loss function was combined with a classified CNN. Furthermore, the fusion layer was embedded in the training process. In this way, CNN can reduce the limitations that were caused by manually developed fusion rules (maximum, minimum, or average).

The use of Image Fusion gives many advantages [27], such as: reduction of data storage and transmission; big possibility of production of a high-resolution output from dark or foggy multiscale images; enhancement of the image from various perspectives that leads to better contrast; and ability to read small signs in different images. However, IF has its limitations too: the feature extraction can sometimes be expensive and complex to perform with fusion. Especially if the images are fuzzy, because it can often lead to very slow data processing.

3.5 Model evaluations methods

After two implementations of the classification model for this course paper, it was necessary to evaluate its quality. We used the following metrics for this

purpose: accuracy, Precision, Recall, F1-score, and ROC curve. These metrics are essential tools for evaluating models' performance [29].

Accuracy is a key ML classification metric that measures the ratio of correct predictions to the total number of predictions [29].

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Toral\ Predictions}$$

Although Accuracy is easy to understand and easy to implement, it does not take into account the nuances of the model, particularly when false positives and false negatives have different consequences.

To overcome accuracy limitations, more sophisticated metrics are needed. F1-score integrates precision and recall, providing a balanced evaluation of classifier performance. It is particularly useful for estimating imbalanced datasets. F1-score components are [29]:

- Precision: the ratio of true positives to all positive predictions. Its formula is: TP / (TP + FP)
- Recall: the ratio of true positives to all actual positive instances. Its formula is: TP / (TP + FN)

And F1-score is calculated by the harmonic mean of precision and recall [29]:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

This formula maintains a balance between precision and recall, with a penalty for extreme values of both. The F1-score ranges from 0 to 1, with 1 being the best result possible.

In the second implementation of the experimental model using Image Fusion, the model performs binary classification. The best evaluation metric for this case is

the ROC curve [30], as it works well for such models. The ROC curve shows the true positive rate compared to the false positive rate at different thresholds.

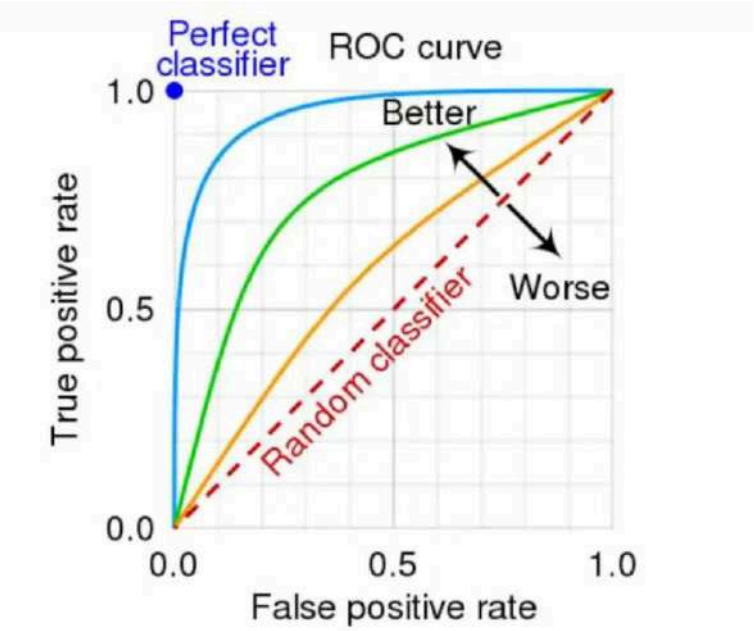

Figure 11. Plot of a ROC curve

Figure 11 showcases the ROC curve view and its meaning. If the ROC curve is closer to the upper left corner, it means the model performs better [30]. The area under the curve (AUC) measures this efficiency in the range from 0 to 1. If AUC value equals 0.5, this indicates a random guess, whereas 1 means a model performs perfectly.

4 EXPERIMENTAL RESEARCH

4.1 Dataset Analysis

For the practical part of the course paper, we have used the “Ukraine Images 2023” dataset from Kaggle [31]. This dataset has 10.4K files and consists of real-world images of damaged buildings by russian shelling from different regions of Ukraine. Most images come from Donetsk (3528 word mentions), Zaporizhzhia (1587), Kharkiv (1585), Kherson (1185), and Dnipropetrovsk (983) regions. In addition to images of damaged buildings, the dataset documents many other consequences of the Russian shelling, including other people, rescuers, ambulances, and even corpses. Despite the fact that such information serves as additional noise for the model and negatively affects its training, we aimed to use this kind of real-world data collection to explore the practical application of the model.

We split the dataset into two parts: train and test. We conduct the experiment in two ways. We outlined five classes:

- Shelling
- Missile
- Drone
- Artillery
- Mine

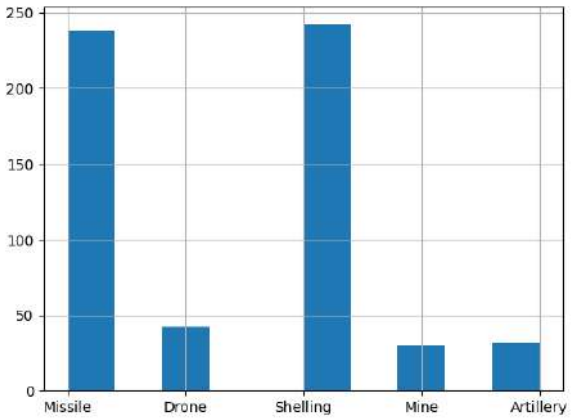

Figure 12. Sample distribution of the 5 classes in “Ukraine Images 2023”

The histogram in Figure 12 shows how imbalanced the class distribution is. We conducted a trial with a more harmonious binary class distribution with partial combination of the previous one:

- Air Attacks (drones, missiles, and rockets)
- Ground attacks (shelling, mines, and artillery)

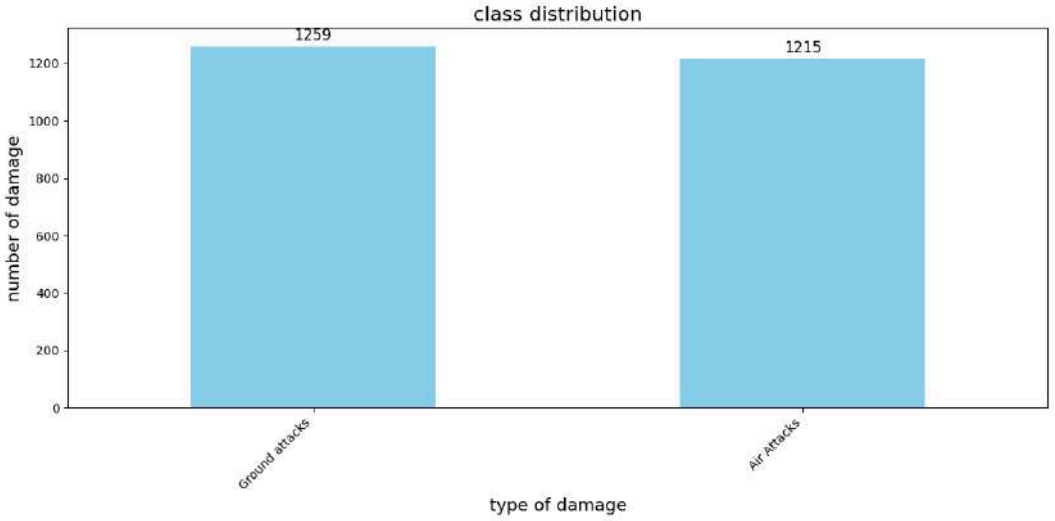

Figure 13. Sample distribution for 2 classes

As can be seen from the histogram in Figure 13, the distribution of binary classes under the combination of attacks is better balanced. The different outcomes of these two distribution methods are described in more detail in the following sections.

4.2 First Experiment: Classification model based on ResNet50V2

To implement the experiment, we have used a transfer learning approach based on a pre-trained ResNet50V2 model, which is an improved version of the ResNet50 architecture. The implementation strategy involves freezing most layers of the base ResNet50V2 model while systematically unfreezing different numbers of terminal layers - specifically, we experimented with unfreezing the final 10, 20, or 30 layers, so that these layers adapt to the specifics of our task. We used fine-tuning [32] in this approach.

After the base model, we have added specialised layers to adapt it more for the damage type classification task:

- GlobalAveragePooling2D: this layer performs feature averaging, which reduces the overfitting of the model and the number of network parameters.
- Dense(128, activation='relu'): It is a fully connected layer ReLU activation for nonlinear feature transformation. L2 regularisation is also applied to prevent overfitting.
- Dropout: during the model training, it accidentally deactivates 40% of neurons, which is used to prevent overfitting too.
- Dense(num_classes, activation='softmax'): this is the output classification layer. To transform the output values into probabilities belonging to each class, the softmax activation is used.

4.3 Second Experiment: Classification Model Using Image Fusion

The particularity of the dataset we chose is that for one event (in this case, a specific case of shelling), there are several different images with damage, which have the same file names, except for the numbering of the photos. It is this special feature of the data that prompted us to choose the Image Fusion method for this task. Using this technique helped us to compensate for the shortcomings of individual images (since the dataset contains real-life content, it has many images in poor lighting and quality) and increase the overall information value of the input data.

Before the classification model with the Image Fusion, a specialised class was developed to perform the work with the data:

1. It handles image sets belonging to the same event.
2. Then it loads these image sets, converting them to a size of 224 x 224, and combines them into a tensor.
3. Finally, it creates a dictionary for mapping class labels to numeric indices.

To directly implement the classification model, we used the pre-trained ResNet18 model as a feature extractor, and we removed its last fully connected layer to obtain the feature vectors. The multi-image fusion mechanism works as follows: each image is processed through a single feature extractor while preserving the group structure for further fusion, and mean pooling is used to aggregate features from different images. The classification layer takes the already aggregated features and generates output logits for each damage type class.

The data processing of the model goes:

- The input is a tensor with a shape (number of damage events in a batch packet, number of images per event, dimension of each image).
- Then the tensor is expanded into a shape for parallel processing of all images.
- The extracted features are then rebuilt back into the form to preserve the group structure.
- The features are aggregated by averaging over the dimensionality of the images to form a single feature vector.
- Finally, these aggregated features are fed to the classification layer to obtain the final predictions.

To train and optimise the model, we used the AdamW optimiser [33] with different training coefficients and the CrossEntropyLoss loss function [34].

5 RESULTS

5.1 First approach results

The results and hyperparameters of the trained model, using the first approach with the ResNet50V2 model, are shown in Table 1:

Mode №	Number of classes	Learning rate	Batch size	Dropout	Training time (in minutes)	Was overfitting?	Additionally	Accuracy
1	5	0.001	8	0.0	19.18	true	Freezing all layers except the last 30 layers of the base model, No Regularisation L2	0.4681
2	5	0.0001	16	0.1	16.03	true	Freezing all layers except the last 20, No Regularisation L2	0.4184
3	5	0.01	32	0.2	31	false	Freezing all layers except the last 10, No Regularisation L2	0.4184
4	5	1e-5	32	0.4	1.27	false	Freezing all layers except the last 10, with Regularisation L2	0.2826
5	5	1e-4	32	0.3	0.39	false	Freezing all layers except the last 10, with Regularisation L2	0.4058
6	5	1e-3	32	0.3	1	true	Freezing all layers except the last 10, with Regularisation L2	0.4755
7	5	5e-6	32	0.5	1	false	Freezing all layers except the last 30, with Regularisation L2	0.4476
8	**2**	**0.001**	**8**	**0.0**	**3.27**	**true**	**Freezing all layers except the last 30, with Regularisation L2**	**0.6626**
9	2	0.0001	16	0.5	1.58	false	Freezing all layers except the last 20, with Regularisation L2	0.6404
10	2	1e-5	32	0.1	1.2	false	Freezing all layers except the last 20, with Regularisation L2	0.4909

11	2	0.01	32	0.4	2.16	false	Freezing all layers except the last 30, with Regularisation L2	0.5091
12	2	5e-6	32	0.4	2.13	false	Freezing all layers except the last 30, with Regularisation L2	0.4909

Table 1. Results of training model with different hyperparameters

As can be seen from the results in Table 1, the model did not achieve high accuracy results. Also, in many cases, the model tended to overfit [35]. It can be concluded that the key problem that affected the results of the experiment was the specificity of the data. The dataset images contain not only various types of damage, but also third-party objects such as rescuers, ambulances, etc. Such a high noise level, different circumstances of the photos, complexity of visual scenes, and an unbalanced amount of data per class negatively affected the model training.

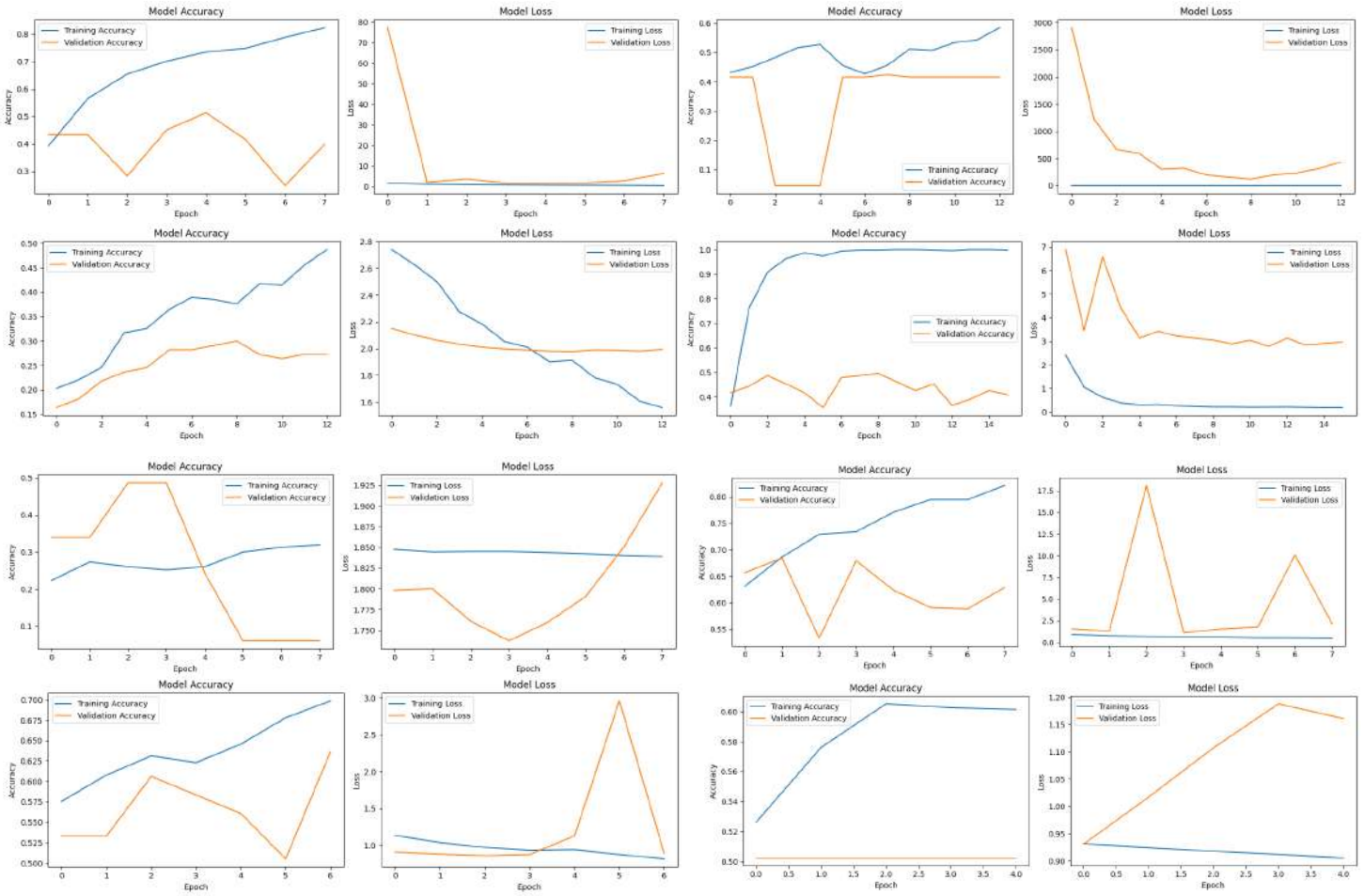

Figure 14. Model's accuracy and loss function dynamics during the training

Figure 14 shows two graphics of the model's accuracy and loss dynamics for each case of training. Most of the graphs show typical signs of overfitting: Training accuracy increases, while test accuracy fluctuates or even decreases. Also, training loss steadily decreases, while test loss often increases after a certain epoch. Most of the other cases show major problems, but the sixth subplot shows the best results among the others, with an accuracy value of 0.6626. Although the test curve has significant swings, the gap between the lines is the smallest compared to the other cases. All these results indicate overall problems in the approach of using the method with the basic architecture of ResNet50V2. The model quickly memorises the features of the training data, but cannot effectively generalise them. This

approach also cannot address high noise levels due to the inability to work with such a specific dataset.

Despite using a powerful architecture such as ResNet50V2, which has demonstrated high performance on many computer vision applications, it was unable to attend to such a complex task. We can identify the following reasons for its poor performance:

- The model is sensitive to noise and tries to find correlations between all elements of the image, including third-party objects that are not related to the type of damage.
- Overfitting due to the limited amount of data for certain classes of damage and many different images, the model tends to remember the features of the training set instead of learning general patterns.
- It is difficult for a model to abstract from context, as it may focus on terrain elements instead of specific damage characteristics.

Thus, classifying the types of damage in real photographs from the war zone is an increasingly difficult task, even for humans, due to the chaotic nature of the destruction. However, if the dataset contains N classes of damage, the expected accuracy of a random classifier is 1/N. Therefore, in our case, for 5 classes, the random accuracy would be 20-25%, so the achieved accuracy of 66% is significantly higher than the baseline.

5.2 Second approach results

The results of the second approach with a model using Image Fusion for two classes (Air and Ground attacks) are shown in Table 2:

Mode №	Learning rate	Batch Size	Accuracy	Precision	Recall	F1-score
1	0.001	16	0.5859	0.6354	0.5859	0.5376
2	**0.01**	**16**	**0.7374**	**0.7522**	**0.7374**	**0.7343**
3	0.01	8	0.6202	0.6265	0.6202	0.6132
4	0.01	32	0.6121	0.6121	0.6121	0.6121
5	0.01	64	0.7010	0.7421	0.7010	0.6895

Table 2. Results of training the second model with different parameters and IF

From the results in Table 2, it can be seen that the model using Image Fusion performs well. This approach significantly improves classification efficiency compared to the first method based on ResNet50V2, with its best results at 73% accuracy. This represents an improvement of 7.48 percent over the previous approach. The high success rate of this IF technique was achieved due to the following:

- A particularity of the dataset used is the presence of several images related to the same event (with a recorded day and a certain place, and an object). The Image Fusion approach made it possible to effectively use this data structure by combining information from different angles and shooting conditions. It also helped to compensate for the flaws of individual images. When real-world data does not have the best quality (noise, poor lighting, blur etc.), IF overcomes these difficulties and provides the model with the maximum possible information from the fused image.
- Processing multiple images simultaneously allowed the model to reduce its sensitivity to context, better abstract from terrain, time of day, and weather conditions. This helped it to focus on the characteristic features of the damage itself.

- The use of the lighter ResNet18 model in combination with the aggregation mechanism provided an effective averaging of features to obtain a more robust representation of the event.

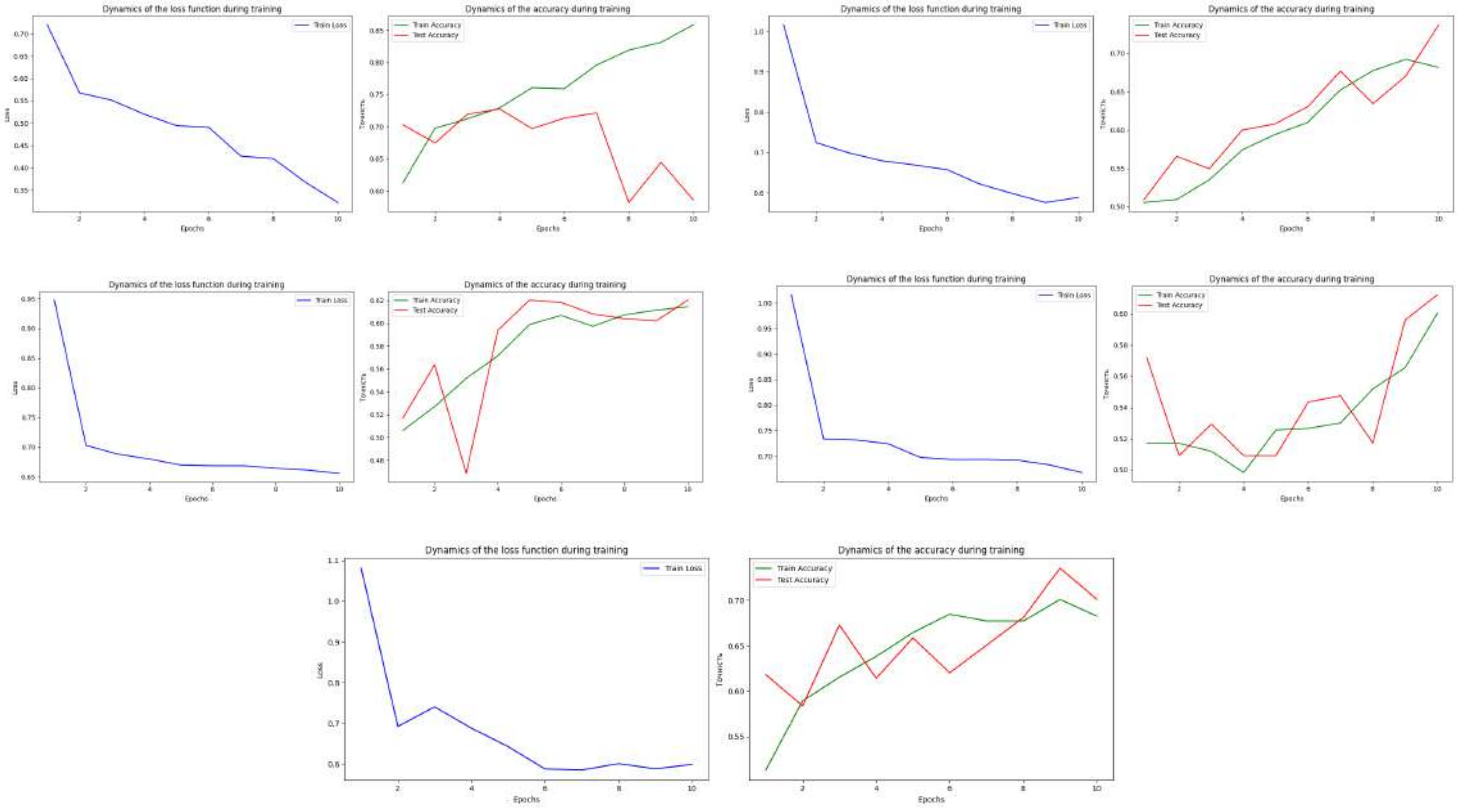

Figure 15. Dynamics of the loss function and accuracy of the model during training

Figure 15 illustrates two subfigures for each case of training: the left one is the dynamics of the loss function, and the right one is the dynamics of accuracy in training and test data. The second case of training demonstrates the most stable and positive dynamics: The loss function graph shows a steady decline from around 0.85 to almost 0.65. The curve decreases smoothly, without sharp swings, indicating a stable learning process; Test accuracy shows an almost steady increase from around 0.5 to 0.7 and above. The training accuracy curve has a similar tendency, demonstrating a good balance. In other cases, there are less ideal patterns: The loss

function decreases but erratically, there is a significant discrepancy between training and test accuracy, and swings in test accuracy indicate instability.

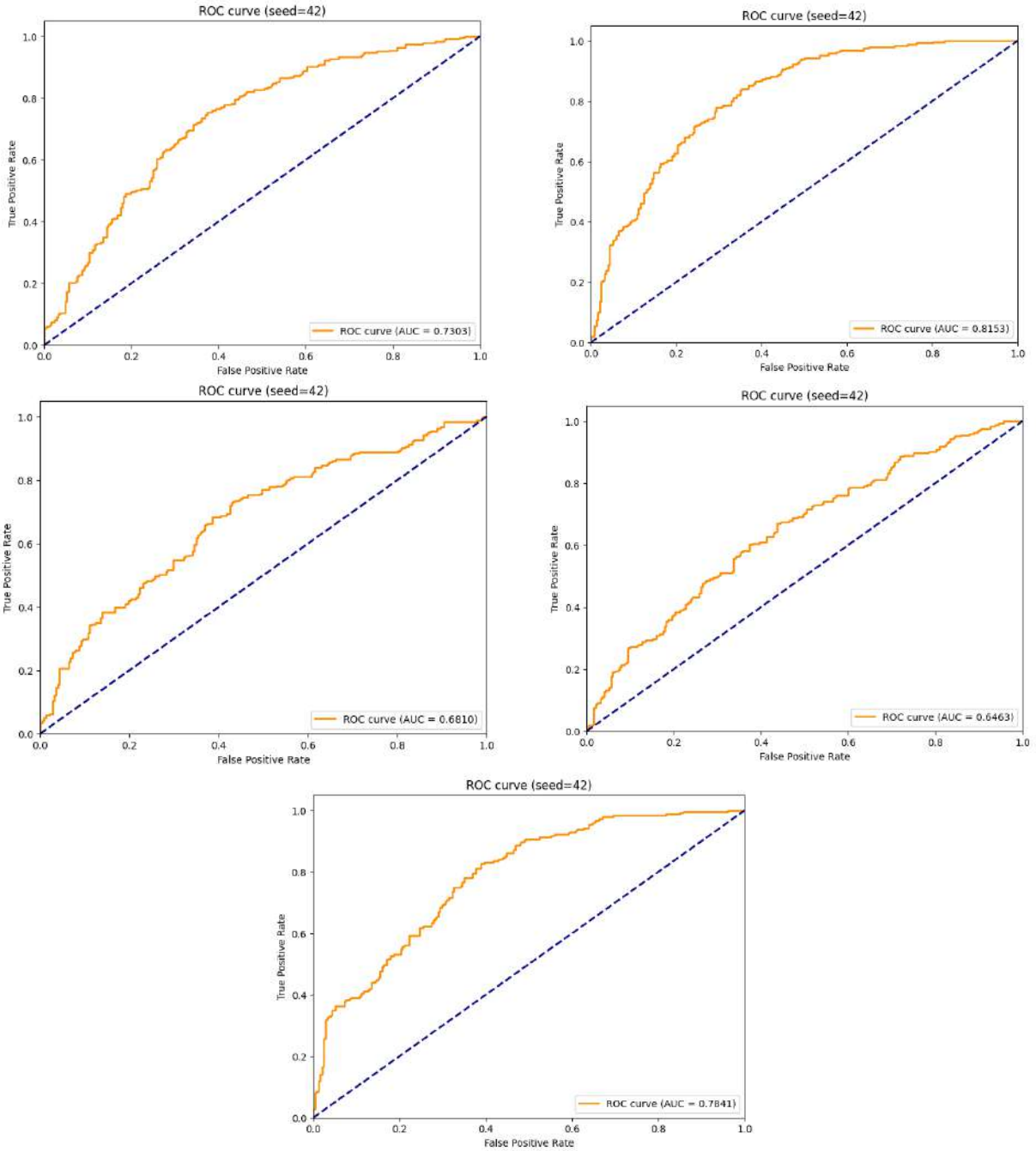

Figure 16. ROC curves for a trained model with different hyperparameters

Figure 16 shows different ROC curves for the corresponding model learning cases. The second graphic demonstrates the smoothest curve with the highest AUC (0.8153). This indicates the model's high ability to distinguish between classes. All other curves are above the diagonal line, which indicates an above-average result, where in all cases the models perform better than the random classifier.

Overall, the achieved classification accuracy of 73.74% has great practical value in the context of the building damage classification task. Such a system can be used to pre-classify a large number of incidents, leaving only questionable cases for manual review. This model has potential for possible future integration into automated shell damage analysis.

6 CONCLUSIONS

The purpose of this course paper is to investigate the problem of classifying building damage caused by military operations in Ukraine using machine learning methods. We analysed the results of two approaches for implementation and concluded which one is better in the applied context. Given the complexity of the problem and the features of the dataset, the achieved classification accuracy is a significant result. The developed models can potentially be used to automate the initial screening of images and support decision-making in assessing the effects of shelling.

Further development of the models may include improving data quality, increasing balancing between classes, and modifying the architecture to include segmentation and attention mechanisms.

REFERENCES

[1] KSE Institute. (2024). Russia Will Pay: Direct Damages Caused to Ukraine's Infrastructure During the Full-Scale War, As of January 1, 2024. Kyiv School of Economics.https://kse.ua/wp-content/uploads/2024/05/Eng_01.01.24_Damages_Report.pdf

[2] Neural network (machine learning). (n.d.). In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

[3] Dougherty, G. (2020). Image classification. In ScienceDirect Topics. https://www.sciencedirect.com/topics/engineering/image-classification

[4] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Insights into Imaging, 9(6), 1–32. https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

[5] Kaur, P., & Agrawal, S. (2020). Image fusion technique. In ScienceDirect Topics. https://www.sciencedirect.com/topics/engineering/image-fusion-technique

[6] Opit. (2023). Computer Vision: How Machines See and Understand the World. https://www.opit.com/magazine/computer-vision-2/

[7] TechTarget. (2024). Deep learning (deep neural network). https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network

[8] Keras. (2024). ResNet and ResNetV2. https://keras.io/api/applications/resnet/

[9] Google Research. (2025). Google Colaboratory. https://colab.research.google.com/

[10] McCormick, C. (2024, April 23). Colab GPUs: Features and Pricing. McCormick ML. https://mccormickml.com/2024/04/23/colab-gpus-features-and-pricing/

[11] Teetor, P. (2024). torchvision: Models, Datasets and Transformations for Images. CRAN. https://cran.r-project.org/web/packages/torchvision/torchvision.pdf

[12] TensorFlow. (2024). tf.keras.applications.ResNet50. TensorFlow API Documentation. https://www.tensorflow.org/api_docs/python/tf/keras/applications/ResNet50

[13] PyTorch. (2025). PyTorch: An open source machine learning framework. https://pytorch.org/

[14] HCL Technologies. (2023). Image Processing vs Deep Learning. Imaging and Machine Vision Europe. https://www.imveurope.com/sites/default/files/content/white-paper/pdfs/HCL_IMVE_WP-ImageProcessing_vs_DL.pdf

[15] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. http://alvarestech.com/temp/deep/Deep%20Learning%20by%20Ian%20Goodfellow,%20Yoshua%20Bengio,%20Aaron%20Courville%20(z-lib.org).pdf

[16] IBM. (2024). Neural Networks. IBM Think. https://www.ibm.com/think/topics/neural-networks

[17] Saha, S. (2023). Convolutional Neural Networks Explained. Towards Data Science. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939/

[18] Levity AI. (2024). How Do Image Classifiers Work? https://levity.ai/blog/how-do-image-classifiers-work

[19] Zhang, X., Zhao, Y., Dong, D., Tan, Z., & Zhang, J. (2024). Max pooling. Scientific Reports, 14. https://www.nature.com/articles/s41598-024-51258-6

[20] Barsagade, S. (2023). ResNet Architecture Explained. Medium. https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d

[21] ThinkingStack. (2024). A Comprehensive Guide to ResNet50 Architecture and Implementation. https://www.thinkingstack.ai/blog/business-use-cases-11/a-comprehensive-guide-to-resnet50-architecture-and-implementation-56

[22] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Google Research. https://static.googleusercontent.com/media/research.google.com/uk//pubs/archive/43442.pdf

[23] Haghighi, M., Pahlavani, P., & Bigdeli, B. (2020). Convolutional block and identity block in residual architecture. ResearchGate. https://www.researchgate.net/figure/Convolutional-block-and-identity-block-in-residual-architecture_fig7_342296550

[24] Dwivedi, P. (2023). ResNets: Residual Blocks & Deep Residual Learning. Towards Data Science. https://towardsdatascience.com/resnets-residual-blocks-deep-residual-learning-a231a0ee73d2/

[25] MathWorks. (2024). ResNet-18 convolutional neural network. https://www.mathworks.com/help/deeplearning/ref/resnet18.html

[26] Product Teacher. (2024). ResNet18 and ResNet50. https://www.productteacher.com/quick-product-tips/resnet18-and-resnet50

[27] ScienceDirect. (2024). Image Fusion. https://www.sciencedirect.com/topics/computer-science/image-fusion

[28] Viso.ai. (2024). Image Fusion in Computer Vision. https://viso.ai/computer-vision/image-fusion-in-computer-vision/

[29] Keylabs. (2024). Understanding the F1 Score and AUC-ROC Curve. https://keylabs.ai/blog/understanding-the-f1-score-and-auc-roc-curve/

[30] Spot Intelligence. (2024, June 17). ROC-AUC Curve in Machine Learning. https://spotintelligence.com/2024/06/17/roc-auc-curve-in-machine-learning/

[31] Nguyen, A. (2023). Ukraine Images 2023. Kaggle. https://www.kaggle.com/datasets/almondnguyen/ukraine-images-2023?select=img_2023

[32] OpenAI. (2024). Fine-tuning. OpenAI Documentation. https://platform.openai.com/docs/guides/fine-tuning

[33] Keras. (2024). AdamW optimizer. https://keras.io/api/optimizers/adamw/

[34] DataCamp. (2024). The Cross-Entropy Loss Function in Machine Learning. https://www.datacamp.com/tutorial/the-cross-entropy-loss-function-in-machine-learning

[35] Google Developers. (2024). Overfitting. Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/overfitting/overfitting?hl=uk

[36] Sharma, V. (2022). Softmax Activation Function: How Does it Actually Work? Pinecone. https://www.pinecone.io/learn/softmax-activation/