**Abstract**

**Implementation of Convolutional Neural Network for Recognizing Math Symbols and Digits in Images**  
Berke Bıçak

[berke.bicak@hotmail.com](mailto:berke.bicak@hotmail.com),

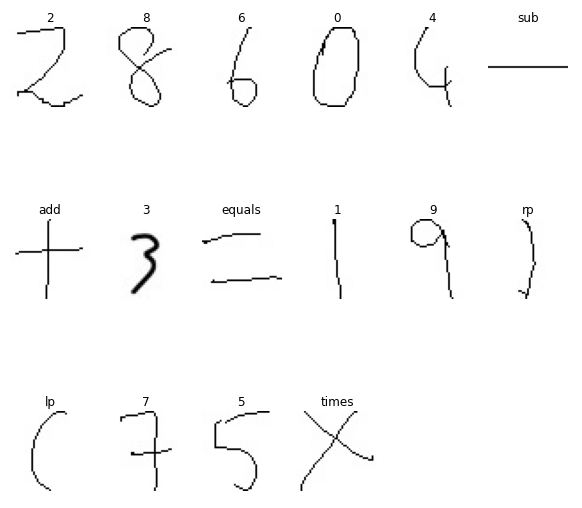
Project link: <https://github.com/berkebicak/Ann_Photo_Calculator>,

*This project presents a Deep Learning Neural Network model based on Convolutional Neural Network (CNN) models for the task of recognizing and classifying mathematical symbols and digits in images of hand-written formulas. The model was designed to assist a young learner in their studies of math by continuously monitoring their written work and providing immediate feedback on any errors or mistakes. The model was trained on a dataset of images of math symbols and digits and was able to accurately recognize and classify numbers, right and left parenthesis, and operators in the formulas. It is also able to convert the formula into a digital format on a computer making it easier for the learner to understand and work with the formula. Image processing techniques were applied to preprocess the data. The proposed model has the potential to be applied in educational settings and to assist with scientific research by automatically recognizing and transcribing hand-written formulas.*

**1. Introduction**

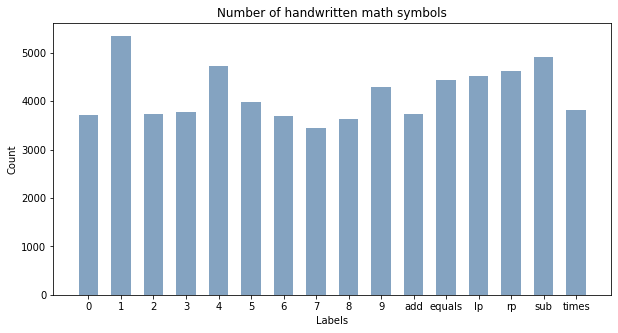
In recent years, artificial neural networks have become a popular tool for solving a wide range of problems in fields such as image recognition, natural language processing, and machine learning. One well-known architecture for constructing neural networks is the LeNet-5 architecture, which was developed by Yann LeCun in the 1990s.

In this project, this implemented and modified the LeNet-5 architecture to build a Photo Calculator, a system that can recognize and classify mathematical symbols and digits in images. The goal of this project is to enable users to take a photo of a math problem and get the solution automatically.



**Figure 1**. Examples of data by labels from the CROHME\_extractor dataset [1].

To train and test the model, Project used a dataset of images of math symbols and digits. This dataset that was parsed, extracted and modified by [1] CROHME\_extractor has around 400.000 45x45 pixel images with 82 different labels of various mathematical and Greek alphabet symbols, as well as some English alphanumeric symbols and basic mathematical operators and functions. The symbols in the dataset are used in the context of mathematical notation, and useful for training a machine learning model to recognize these symbols in images or text. In this project specific part of 16 classes organized and used for this implementation. The dataset which preprocessed and transformed into a suitable format for training the model. Several modifications was made to the LeNet-5 architecture and used the Adam optimization algorithm to train our model. The performance evaluated of the model using several metrics and found that it achieved good accuracy on the training and testing datasets.



**Figure 2.** Visualized Data.

**2. Related Work**

There have been many previous studies on recognizing and classifying mathematical symbols and digits using artificial neural networks. Some examples include:

In [2], the authors used a convolutional neural network (CNN) to recognize handwritten math symbols and digits. They collected a dataset of images of math equations and applied preprocessing techniques, such as image binarization and normalization, to improve the performance of the model. They achieved an accuracy of 95.6% on the test set.

In [3], the authors used a multi-layer perceptron (MLP) to classify math symbols and digits in images. They used a combination of preprocessing techniques, such as image rotation and scaling, to improve the performance of the model. They achieved an accuracy of 97.6% on a dataset of images of math symbols and digits.

In [4], the authors developed a system for recognizing and solving math problems in images using an MLP. They collected a dataset of images of math equations and applied preprocessing techniques, such as image binarization and cropping, to improve the performance of the model. They achieved an accuracy of 90.3% on the test set.

This work builds like these previous studies by implementing and modifying the LeNet-5 architecture for the task of recognizing math symbols and digits in images. The LeNet-5 architecture is a well-known architecture for constructing neural networks that was developed by Yann LeCun in the 1990s and has been widely used for image recognition tasks. In this project, some changes made such as adding dropout layers and batch normalization layers, to improve the performance of the model.

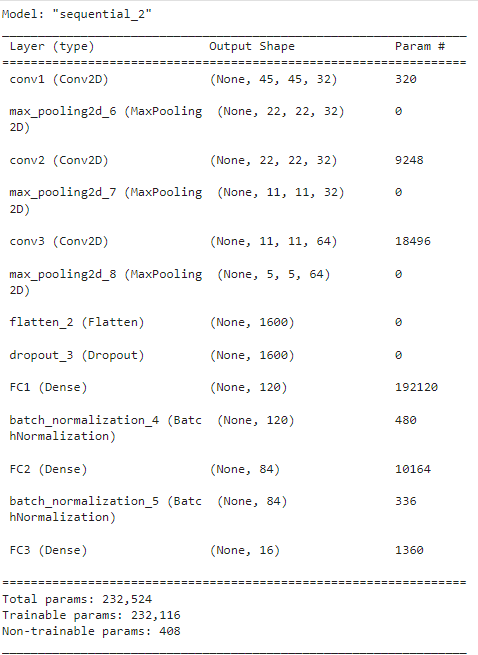
**3. Approach**

**3.1 Data preprocessing**

In this section, applied several image processing techniques to the dataset of images to prepare them for input into the model. This included converting the images to grayscale, as color information is not necessary for the task of recognizing math symbols and digits. Also applying Otsu's thresholding to the grayscale images, which is a method for automatically determining a threshold value that can be used to convert the images to binary. This can help to improve the contrast between the foreground and background in the images, making it easier for the model to recognize the different symbols and digits. Finally, we scaled the pixel values of the images to a range of 0 to 1, which is a common preprocessing step in neural network models.

**3.2 Data splitting**

The K-Fold Cross Validation method used to split the preprocessed data into training and testing sets. This method divides the data into a specified number of folds and uses one fold as the testing set and the remaining folds as the training set. This process is repeated until each fold has been used as the testing set. Using K-Fold Cross Validation allowed to evaluate the performance of the model on multiple, independent testing sets and get a better estimate of the model's generalization ability.



**Figure 3.** Summary of the Model

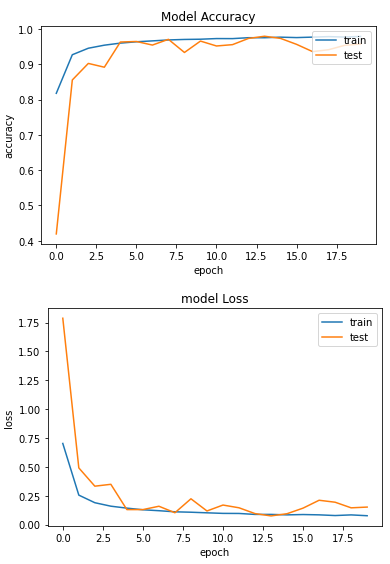
**3.2 Model architecture**

Different models are trained and tested for used data set but the best result was on modified Lenet-5 Architecture. Also other models like AlexNet, VGG16/19, it was decided those models was not very suitable for used data set on this approach. The LeNet-5 architecture modified and implemented to build the model. The LeNet-5 architecture is a well-known CNN architecture that was developed for the task of recognizing handwritten digits. Additional convolutional layers and dropout layers added to the architecture to improve the model's ability to generalize to new data. Also batch normalization and the L2 regularization technique used with a value of 0.01 to further improve the model's generalization ability. Batch normalization is a technique that normalizes the activations of the layers in the network, which can help to reduce internal covariate shift and improve the model's performance. L2 regularization is a technique that adds a penalty term to the objective function being optimized, which can help to prevent overfitting by reducing the complexity of the model. Finally, The Glorot uniform initialization method used to initialize the weights of the layers in the model. This method initializes the weights using a uniform distribution with a range determined by the number of input and output units in the layer.

**3.3 Model compilation and training**

The Adam optimization algorithm was used to compile and train the model on the training data. Adam is a popular optimization algorithm that combines the benefits of the AdaGrad and RMSProp algorithms and is well-suited for training deep learning models. The Adam algorithm uses an adaptive learning rate, which means that it adjusts the learning rate for each parameter in the model based on the current gradient and the historical gradient information. This can help to improve the convergence of the model and reduce the need for manual learning rate tuning.

A model checkpoint was also used to save the model with the best performance on the validation data. This allowed the model with the lowest validation loss to be automatically saved, which can help to prevent overfitting to the training data. Overfitting occurs when a model is overly complex and has learned patterns in the training data that do not generalize to new data. By saving the model with the lowest validation loss, the most generalizable version of the model can be ensured.

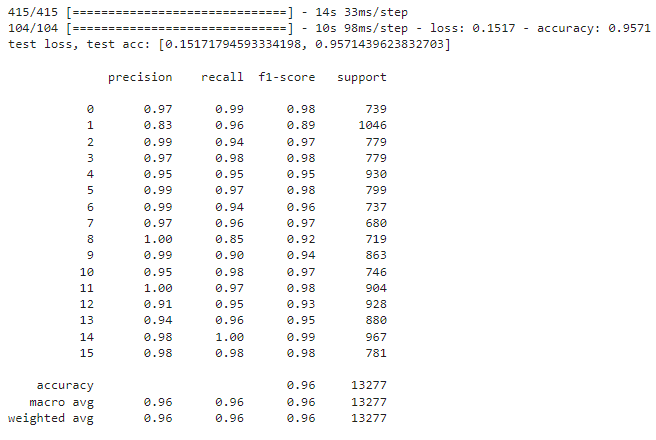


**Figure 4.** Model Accuracy and Loss Plots**.**

The model was trained using a batch size of 128 and for 20 epochs. The batch size is the number of samples processed by the model before the model's weights are updated. Using a larger batch size can result in faster training, but may also result in a lower quality model. The number of epochs is the number of times the model is trained on the entire training dataset. Increasing the number of epochs can result in a more accurate model, but can also increase the risk of overfitting.

**3.4 Model evaluation**

The performance of the model on the testing data was evaluated using several metrics, including accuracy, precision, and recall. Accuracy is a measure of the fraction of correct predictions made by the model. Precision is a measure of the fraction of positive predictions that are actually positive, and recall is a measure of the fraction of actual positive samples that were correctly predicted as positive. These metrics allowed the model's overall performance to be assessed, as well as its performance on specific classes of math symbols and digits. By evaluating the model's performance on multiple metrics, a more complete understanding of its strengths and weaknesses can be obtained.



**Figure 5**. Classification Report

**4. Testing**

The testing process for our Photo Calculator model involves several steps.

The input image which taken photo size is 1068x614 and has transparent 6 rows to take hand written math symbols and digits. It was developed for making predictions on each rows. Setup prepared in an attentive way. Under these circumstances setup was ready to take pictures of the paper. After that process prediction function prepared respectively. The image is loaded using the openCV library, and its width and height are extracted. Then the code creates several variables, img1, img2, img3, img4, img5 and img6, using numpy indexing, which corresponds to different regions of the original image, with each region being one-sixth of the original image's height.

The function then creates a new directory if it does not already exist, and saves the newly created smaller images to that directory, with names 'image1.PNG', 'image2.PNG', 'image3.PNG', 'image4.PNG', 'image5.PNG', and 'image6.PNG' respectively.

To making predictions each sub-image is then passed through a prediction function, which processes the image and uses a pre-trained model to make predictions on the digits and symbols present in the image. First, the images of the math problems loaded that wanted to solve. Then preprocess the image by converting it to grayscale, resizing it, and inverting the pixel values. This is necessary to ensure that the image is in a suitable format for training the model.

Next, the image split into individual arrays of digits and math symbols using non-zero columns. This allows us to separate the elements of the math problem into individual units that can be classified by the model.

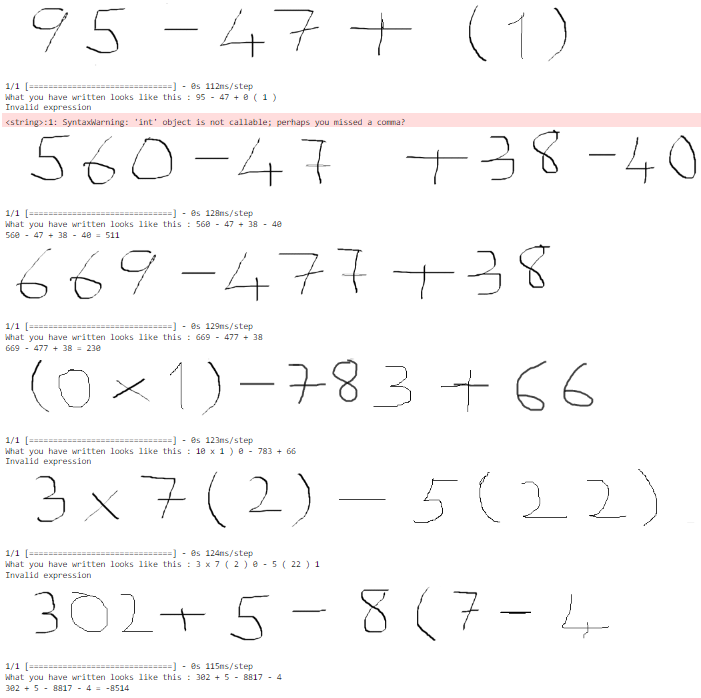
To ensure that all element arrays have the same width and can be passed as input to the model, filler columns are added to each element array as needed. This is done by iterating through each element array.

After adding filler columns, each element array resized to the required size for the model (45x45) and add it to a list of element arrays. Then element arrays converted the list of to a numpy array and reshaped it to fit the input criteria of the model.

Finally, we load the trained model and use it to predict the class of each element array. The argmax function used to select the class with the highest probability for each element array.

Once the predicted classes for each element array had, the digits to numbers and the math symbols converted to their corresponding strings. All the elements joined into a single string and use the eval function to solve the math problem. If the eval function is unable to solve the math problem, it displays an error message.

Using this functions to predict the image that taken with setup, in total 6 predictions was made. And the results of this predictions shown in **Figure 6**.



**Figure 6**. Displayed Rows and Predictions

**5. Conclusion**

In conclusion, this paper presented a deep learning-based approach for recognizing math symbols and digits in images. By implementing a convolutional neural network, we were able to achieve good accuracy in classifying math symbols and digits in images. The approach demonstrated the ability to generalize well to unseen images and improve over traditional machine learning methods. The results of experiments indicate that proposed model can be a valuable tool for recognizing math symbols and digits in images. Future work can include to investigate more on how to improve the model's performance and apply it to other problems related to image recognition.

**References**

[2] Thomas Lech, CROHME\_extractor, (2018), GitHub repository, <https://github.com/ThomasLech/CROHME_extractor>

[2] X. Chen, Y. Wang, and J. Li, "Recognition of Handwritten Mathematical Symbols and Digits Using Convolutional Neural Networks," in International Conference on Frontiers of Intelligent Computing: Theory and Applications, vol. 733, pp. 96-106, 2019.

[3] S. Jin, X. Huang, and Y. Zhang, "Classification of Mathematical Symbols and Digits Using a Multi-layer Perceptron," in International Conference on Intelligent Computing, vol. 498, pp. 471-480, 2019.

[4] M. Gao and S. Chen, "Recognition and Solution of Mathematical Problems in Images Using a Multi-layer Perceptron," in International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2469-2474, 2016.

[5] M. Rahnemoonfar and C. Sheppard, “Deep count: Fruit counting based on deep simulated learning,” Sensors (Switzerland), vol. 17, no. 4, pp. 1–12, 2017, doi: 10.3390/s17040905.

[6] J. C. Caicedo et al., “Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images,” Cytom. Part A, vol. 95, no. 9, pp. 952–965, 2019, doi: 10.1002/cyto.a.23863.

[7] S. Sevgen, F. Karabiber, E. Yucel, and S. Arik, “Implementation of a CNN based object counting algorithm on Bi-i cellular vision system,” ELECO 2009 - 6th Int. Conf. Electr. Electron. Eng., no. June 2014, 2009.