Full depth CNN classifier for handwritten and license plate characters recognition

Mohammed Salemdeeb [Corresp., 1], **Sarp Ertürk** [2]

[1] Dep. of Electrical-Electronics Engineering, Bartin University, Bartin, Turkey

[2] Dep. of Electronics & Communication Eng., Kocaeli University, Izmit, Kocaeli, Turkey

Corresponding Author: Mohammed Salemdeeb
Email address: msalemdeeb@bartin.edu.tr

Character recognition is an important research field of interest for many applications. In recent years, deep learning has made breakthroughs in image classification, especially for character recognition. However, convolutional neural networks (CNN) still deliver state-of-the-art results in this area. Motivated by the success of CNNs, this paper proposes a simple novel full depth stacked CNN architecture for Latin and Arabic handwritten alphanumeric characters that is also utilized for license plate (LP) characters recognition. The proposed architecture is constructed by four convolutional layers, two max-pooling layers, and one fully connected layer. This architecture is low-complex, fast, reliable and achieves very promising classification accuracy that may move the field forward in terms of low complexity, high accuracy and full feature extraction. The proposed approach is tested on four benchmarks for handwritten character datasets, Fashion-MNIST dataset, public LP character datasets and a newly introduced real LP isolated character dataset. The proposed approach tests report an error of only 0.28% for MNIST, 0.34% for MAHDB, 1.45% for AHCD, 3.81% for AIA9K, 5.00% for Fashion-MNIST, 0.26% for Saudi license plate character and 0.97% for Latin license plate characters datasets. The license plate characters include license plates from Turkey (TR), Europe (EU), USA, United Arab Emirates (UAE) and Kingdom of Saudi Arabia (KSA).

Full Depth CNN Classifier for Handwritten and License Plate Characters Recognition

Mohammed Salemdeeb[1] **and Sarp Ertürk**[2]

[1]**Dep. of Electrical-Electronics Engineering, Bartin University, Bartin, Turkey.**
[2]**Dep. of Electronics & Communication Eng., Kocaeli University, Izmit, Kocaeli, Turkey**

Corresponding author:
Mohammed Salemdeeb[1]

Email address: msalemdeeb@bartin.edu.tr

ABSTRACT

Character recognition is an important research field of interest for many applications. In recent years, deep learning has made breakthroughs in image classification, especially for character recognition. However, convolutional neural networks (CNN) still deliver state-of-the-art results in this area. Motivated by the success of CNNs, this paper proposes a simple novel full depth stacked CNN architecture for Latin and Arabic handwritten alphanumeric characters that is also utilized for license plate (LP) characters recognition. The proposed architecture is constructed by four convolutional layers, two max-pooling layers, and one fully connected layer. This architecture is low-complex, fast, reliable and achieves very promising classification accuracy that may move the field forward in terms of low complexity, high accuracy and full feature extraction. The proposed approach is tested on four benchmarks for handwritten character datasets, Fashion-MNIST dataset, public LP character datasets and a newly introduced real LP isolated character dataset. The proposed approach tests report an error of only 0.28% for MNIST, 0.34% for MAHDB, 1.45% for AHCD, 3.81% for AIA9K, 5.00% for Fashion-MNIST, 0.26% for Saudi license plate character and 0.97% for Latin license plate character datasets. The license plate characters include license plates from Turkey (TR), Europe (EU), USA, United Arab Emirates (UAE) and Kingdom of Saudi Arabia (KSA).

INTRODUCTION

Character recognition (CR) plays a key role in many applications and motivates R&D in the field for accurate and fast classification solutions. CR has been widely investigated in many languages using different proposed methods. In the last years, researchers widely used CNN as deep learning classifiers and achieved good results on handwritten Alphanumeric in many languages (Lecun et al., 1998; Abdleazeem and El-Sherif, 2008; El-Sawy et al., 2017; Guha et al., 2020), character recognition in real-world images (Netzer et al., 2011), document scanning, optical character recognition (OCR) and automatic license plate character recognition (ALPR) (Comelli et al., 1995). Searching for text information in images is a time-consuming process that largely benefits of CR. Particularly, in Arabic language the connectivity of letters make a challenge for classification (Eltay et al., 2020). Therefore, isolated character datasets get more interest in research.

MNIST is a handwritten digits dataset introduced by Lecun et al. (1998) and used to test supervised machine learning algorithms . The best accuracy obtained by stacked CNN architectures, until before two years, is a test error rate of 0.35% in (Cireşan et al., 2010), where large deep CNN of nine layers with an elastic distortion applied to the input images. Narrowing the gap to human performance, a new architecture of five committees of seven deep CNNs with six width normalization and elastic distortion was trained and tested in (Ciresan et al., 2011) and reported an error rate of 0.27%, where the main CNN is seven stacked layers. In (Ciregan et al., 2012), a near-human performance error rate of 0.23% was achieved, where several techniques were combined in a novel way to build a multi-column deep neural network (MCDNN) inspired by micro-columns of neurons in cerebral cortex compared to the number of layers found between retina and visual cortex of macaque monkeys.

Recently, Moradi et al. (2019) developed a new CNN architecture with orthogonal feature maps based

on Residual modules of ResNet (He et al., 2016) and Inception modules of GoogleNet (Szegedy et al., 2015), with 534474 learnable parameters which are equal to SqeezeNet (Iandola et al., 2016) learnable parameters, and thus the model reported an error of 0.28%. However, a CNN architecture for small size input images of 20×20 pixels was proposed in (Le and Nguyen, 2019). In addition, a multimodal deep learning architecture was proposed in (Kowsari et al., 2018), where deep neural networks (DNN), CNN and recurrent neural networks (RNN) were used in one architecture design achieving an error of 0.18%. A plain CNN with stochastic optimization method was proposed in (Assiri, 2019), this method applied regular Dropout layers after each pooling and fully connected (FC) layers, this 15 stacked layers approach obtained an error of 0.17% by 13.21M parameters. Hirata and Takahashi (2020) proposed an architecture with one base CNN and multiple FC sub-networks, this 28 spars layers architecture with 28.67M parameters obtained an error of 0.16%. Byerly et al. (2020) presented a CNN design with additional branches after certain convolutions, and from each branch, they transformed each of the final filters into a pair of homogeneous vector capsules, this 21 spars layers obtained an error of 0.16%.

While MNIST was well studied in the literature, there were only a few works on Arabic handwritten character recognition (Abdleazeem and El-Sherif, 2008). The large Arabic Handwritten Digits (AHDBase) has been introduced in (El-Sherif and Abdelazeem, 2007). Abdleazeem and El-Sherif (2008) modified AHDBase to be MADBase and evaluated 54 different classifier/features combinations and reported a classification error of 0.52% utilizing radial basis function (RBF) and support vector machine (SVM). Also, they discussed the problem of Arabic zero, which is just a dot and smaller than other digits.They solved the problem by introducing a size-sensitive feature which is the ratio of the digit bounding box area to the average bounding box area of all digits in AHDBase's training set. In the same context, Mudhsh and Almodfer (2017) obtained a validation error of 0.34% on the MADBase dataset by using an Alphanumeric VGG network inspired by the VGGNet (Simonyan and Zisserman, 2015) with dropout regularization and data augmentation but the error performance does not hold on the test set.

Torki et al. (2014) introduced AIA9K dataset and reported a classification error of 5.72% on the test set by using window-based descriptors with some common classifiers such as logistic regression, linear SVM , nonlinear SVM and artificial neural networks (ANN) classifiers. Younis (2017) tested a CNN architecture and obtained an error of 5.2%, he proposed a stacked CNN of three convolution layers followed by batch normalization, rectified linear units (ReLU) activation, dropout and two FC layers.

The AHCD dataset was introduced by El-Sawy et al. (2017), they reported a classification error of 5.1% using a stacked CNN of two convolution layers, two pooling layers and two FC layers. Najadat et al. (2019) obtained a classification error of 2.8% by using a series CNN of four convolution layers activated by ReLU, two pooling layers and three FC layers. The state-of-the-art result for this dataset is a classification error of 1.58% obtained by Sousa (2018), it was achieved by ensemble averaging of four CNNs, two inspired by VGG16 and two written from scratch, with batch normalization and dropout regularization, to form 12 layers architecture called VGG12.

For benchmarking machine learning algorithms on tiny grayscale images other than Alphanumeric characters, Xiao et al. (2017) introduced Fashion-MNIST dataset to serve as a direct replacement for the original MNIST dataset and reported a classification test error of 10.3% using SVM. This dataset gained the attention of many researchers to test their approaches and better error of 3.65% was achieved by Zhong et al. (2017) in which a random erasing augmentation was used with wide residual networks (WRN) (Zagoruyko and Komodakis, 2016). The state-of-the-art performance for Fashion-MNIST is an error of 2.34% reported in (Zeng et al., 2018) using a deep collaborative weight-based classification method based on VGG16. Recently, a modelling and optimization based method was used (Chou et al., 2019) to optimize the parameters for a multi-layer (16 layer) CNN reporting an error of 8.32% and 0.57% for Fashion-MNIST and MNIST respectively.

ALPR is a group of techniques that use CR modules to recognize vehicle's LP number. Sometimes, it is also referred to as license plate detection and recognition (LPDR). ALPR is used in many real-life applications (Du et al., 2013) like electronic toll collection, traffic control, security, etc. The main challenges of detection and recognition of license plates are the variations in the plate types, environments, languages and fonts. Both CNN and traditional approaches are used to solve vehicle license plates recognition problems. Traditional approaches involve computer vision, image processing and pattern recognition algorithms for features such as color, edge and morphology (Xie et al., 2018). A typical ALPR system consists of three modules, plate detection, character segmentation and CR modules (Shyang-Lih Chang et al., 2004). This research focuses on CR techniques and compared them with the proposed CR

approach. CR modules need an off-line training phase to train a classifier on each isolated character using a set of manually cropped character images (Bulan et al., 2017). Excessive operational time, cost and efforts must be considered when manual cropping of character images are needed to be collected and labeled for training and testing, and to overcome this, artificially generated synthetic license plates were proposed (Bulan et al., 2015).

Additionally, very little research was done on multi-language LP character recognition, the reason is mostly due to the lack of multi-language LP datasets. Some recent researches were interested in introducing a global ALPR system. Asif et al. (2017) studied only LP detection module using a histogram-based approach, and a private dataset was used, which comprised of LPs from Hungary, America, Serbia, Pakistan, Italy, China, and UAE (Asif et al., 2017). VGG and LSTM were proposed for CR module in (Dorbe et al., 2018) and the measured CR module accuracy was 96.7% where the test was done on LPs from Russia, Poland, Latvia, Belarus, Estonia, Germany, Lithuania, Finland and Sweden. Also, tiny YOLOv3 was used as a unified CR module for LPs from Greece, USA, Croatia, Taiwan, and South Korea (Henry et al., 2020). Furthermore, several proposed methods interested in multi-language LPCR testing CR modules on each LP country's dataset separately, without accumulating the characters into one dataset (Li et al., 2019; Yépez et al., 2019; Asif et al., 2019). In addition, Selmi et al. (2020) proposed a mask R-CNN detector for character segmentation and recognition concerning Arabic and English LP characters from Tunisia and USA. Park et al. (2019) concerned USA and Korean LPs describing the problem as multi-style detection. CNN shrinkage-based architecture was studied in (Salemdeeb and Erturk, 2020), utilizing the maximum number of convolutional layers that can be added. Salemdeeb and Erturk (2020) studied the LP detection and country classification problem for multinational and multi-language LPs from Turkey, Europe, USA, UAE and KSA, without studying CR problem. These researches studied LPs from 23 different countries where most of them use Latin characters to write the LP number, and totally five languages were concerned (English, Taiwanese, Korean, Chinese and Arabic). In Taiwan, Korea, China, UAE, Tunisia and KSA, the LP number is written using Latin characters, but the city information is coded using characters from that the country's language.

In this paper, Arabic and Latin isolated characters are targeted to be recognized using a proposed full depth CNN (FDCNN) architecture in which the regions of interest are USA, EU and Middle East. To verify the performance of the proposed FDCNN, some isolated handwritten Arabic and Latin characters benchmarks such as MNIST, MADbase, AHCD, AIA9K datasets are also tested. Also, a new dataset named LP Arabic and Latin isolated characters (LPALIC) is introduced and tested. In addition, the recent FashionMNIST dataset is also tested to generalize the full depth feature extraction approach performance on tiny grayscale images. The proposed FDCNN approach closes the gap between software and hardware implementation since it provides low complexity and high performance. All the trained models and the LPALIC dataset[1] are made publicly available online for research community and future tests.

The rest of this paper is organized as follows; section 2 introduces the structure of datasets used in this paper and also the new LPALIC dataset. In section 3, the proposed approach is described in details. Section 4 presents a series of experimental results and discussions. Finally, section 5 summarizes the main points of the entire work as a conclusion.

DATASETS

Datasets Available in the Literature

MNIST is a low-complexity data collection of handwritten digits to test supervised machine learning algorithms introduced by Lecun et al. (1998). It has grayscale images of size 28×28 pixels with 60000 training digits and 10000 test digits written by different persons. The digits are white and have black background, normalized to 20×20 pixels preserving the aspect ratio, and then centered at the center of mass of the 28×28 pixels grayscale images. The official site for the dataset and results are availabe by LeCun[2].

In MADbase, 700 Arabic native writers wrote ten digits ten times and the images were collected as 70000 binary images; 60000 for training and 10000 for testing, so that writers of training set and test set are exclusive. This dataset[3] has the same format as MNIST to make veracity for comparisons between

[1] https://www.kaggle.com/dataset/b4697afbddab933081344d1bed3f7907f0b2b2522f637adf15a5fcea67af2145
[2] http://yann.lecun.com/exdb/mnist/
[3] http://datacenter.aucegypt.edu/shazeem

digits (used in Arabic and English languages) recognition approaches. Table 1 shows example digits of printed Latin, Arabic and handwritten Arabic characters used for numbers as declared in ISO/IEC 8859-6:1999.

Table 1. Printed and handwritten digits.

Printed Latin Characters	0	1	2	3	4	5	6	7	8	9
Printed Arabic Characters	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Handwritten										

AHCD dataset[4] consists of 13440 training images and 3360 test images for 28 Arabic handwritten letters (classes) of size 32×32 pixels grayscale images. In AI9IK[5] dataset, 62 female and 45 male Arabic native writers aged between 18 to 25 years old at the Faculty of Engineering at Alexandria University-Egypt were invited to write all the Arabic letters 3 times to gather 8988 letters of which 8737 32×32 grayscale letter images were accepted after a verification process by eliminating cropping errors, writer mistakes and unclear letters. FashionMNIST dataset[6] has images of 70000 unique products taken by professional photographers. The thumbnails (51×73) were then converted to 28×28 grayscale images by the conversion pipeline declared in (Xiao et al., 2017). It is composed of 60000 training images and 10000 test images of 10 class labels.

Table 2 gives a brief review on some publicly available related LP datasets for LPDR problem. The Zemris dataset is also called English LP in some references (Panahi and Gholampour, 2017).

Table 2. A review of publicly available ALPR datasets.

Dataset	Approach	Number of Images	Accuracy	Classifier	Character Set	Purpose
Zemris	Kraupner (2003)	510	86.2%	SVM	No	LPDR
UCSD	Dlagnekov (2005)	405	89.5%	OCR	No	LPDR
Snapshots	Martinsky (2007)	97	85%	MLP	No	LPDR
ARG	Fernández et al. (2011)	730	95.8%	SVM	No	LPDR
SSIG	Gonçalves et al. (2016)	2000	95.8%	SVM-OCR	Yes	LPDR
ReId	Špaňhel et al. (2017)	77k	96.5%	CNN	No	LPR
UFPR	Laroca et al. (2018)	4500	78.33%	CR-NET	Yes	LPDR
CCPD	Xu et al. (2018)	250k	95.2%	RPnet	Yes	LPDR

Novel License Plate Characters Dataset

This research introduces a new multi-language LP chatacters dataset, involving both Latin and Arabic characters from LP images used in Turkey, USA, UAE, KSA and EU (Croatia, Greece, Czech, France, Germany, Serbia, Netherlands and Belgium). It is called LPALIC datase. In addition, some characters cropped from Brazil, India and other countries were added for just training to give features diversity. Furthermore, Some characters were collected from some public LP datasets, LP websites and our own camera pictures in Turkey taken in different weather conditions, places, blurring, distances, tilts and illuminations. These characters are real LP manually cropped characters without any filtering. For uniformity a size of 28×28 pixels of grayscale images was utilized.

The manually cropped characters were fed into the following conversion pipeline inspired from FashionMNIST (Xiao et al., 2017) which is similar to MNIST (Lecun et al., 1998) ,

1. Resizing the longest edge of the image to 24 to save the aspect ratio.
2. Converting the image to 8-bit grayscale pixels image.

[4]https://www.kaggle.com/mloey1/ahcd1
[5]www.eng.alexu.edu.eg/%7emehussein/AIA9k/index.html
[6]github.com/zalandoresearch/fashion-mnist

3. Negating the intensities of the image to get white character with black background.
4. Computing the center of mass of the pixels.
5. Translating the image to put center of mass at the center of the 28×28 grayscale image.

Some samples of the LPALIC dataset is visualized in Figure 1 for Latin characters and in Figure 2 for Arabic characters.

Figure 1. Samples of Latin characters in the LPALIC dataset.

Characters ”0” and ”O” are in the same class label so Latin characters have 35 (10 digits and 25 letters) class labels and Arabic characters have 27 (10 digits and 17 letters) class labels as LP as used in KSA. Table 3 illustrates the total number of Arabic and Latin characters included in LPALIC dataset.

Table 3. LPALIC dataset number of cropped characters per country.

Country	TR	EU	USA	UAE	Others	KSA
Used Characters	Latin	Latin	Latin	Latin & Arabic	Latin	Arabic
Number of Characters	60000	32776	7384	3003	17613	50000
Total Characters	120776					50000

The Latin characters were collected from 11 countries (LPs have different background and font colors) while the Arabic characters were collected from only KSA (LPs have white background and black character). Choosing those countries is related to the availability of those LPs for public use.

PROPOSED APPROACH

Stacked CNN architecture is simple, where each layer has a single input and a single output. For small size images, the key efficient simple deep learning architecture was LeNet-5 (Lecun et al., 1998), it consists of

Figure 2. Samples of Arabic characters in the LPALIC dataset.

three convolutional, two pooling and one FC (Dense) layer. It was used and developed for the models in (Cireşan et al., 2010) and in the main column of MCDNN in (Ciregan et al., 2012). Most of recent architectures are sparse structure of CNN such as models in GoogleNet (Szegedy et al., 2015), ResNet (He et al., 2016) and DensNet (Huang et al., 2017).

Proposed Architecture

The core of the proposed model is the convolution block which is a convolutional layer followed by a batch normalization (BN) layer (Ioffe and Szegedy, 2015) and a non-linear activation ReLU layer (Krizhevsky et al., 2012). This block is called standard convolutional layer in (Howard et al., 2017). The proposed convolutional layers have kernels of size 5×5 with a single stride. This kernel size showed a good feature extraction capability in LeNet-5 (Lecun et al., 1998) for small images as it covers 3.2% of the input image in every stride. However, the recent trends are to replace 5×5 with 2 layers of 3×3 kernels as in InceptionV3 (Szegedy et al., 2016). Figure 3 shows the architecture design of the proposed model.

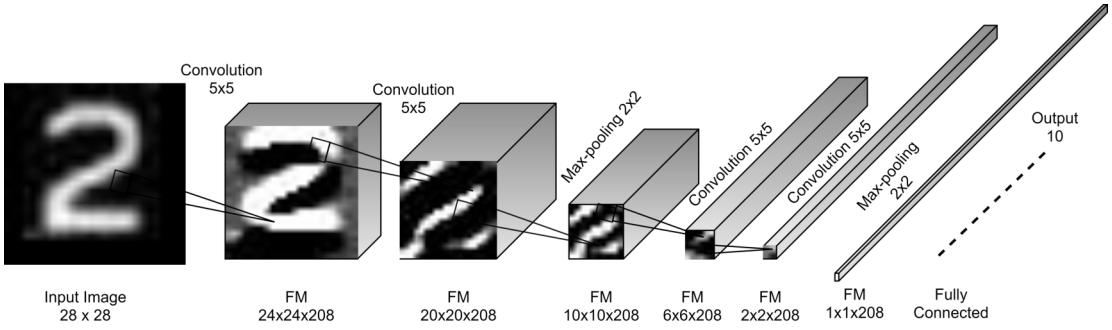


Figure 3. Proposed FDCNN model architecture.

BN layer was recently introduced by Ioffe and Szegedy (2015). It normalizes the input by subtracting the mean of batch and dividing by the batch standard deviation then it scales and shifts the normalized input by learnable scale and shift, it reduces covariance shift, reduces overfitting, enables higher learning

rates, regularizes the model and fulfills some of the same goals as Dropout layers. The first designers used BN layer in InceptionV3 are Szegedy et al. (2016).

For a mini-batch $B = \{x_1, x_2, ..., x_m\}$ of size m, the mean μ_B and variance σ_B^2 of B is computed and each input image in the mini-batch is normalized according to Equation (1).

$$\hat{x_i} = \frac{(x_i) - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{1}$$

Where ε is a constant, $\hat{x_i}$ is the i^{th} normalized image scaled by learnable scale parameter γ and shifted by learnable shift parameter β producing the i^{th} normalized output image y_i (Ioffe and Szegedy, 2015).

$$y_i = BN_{\gamma,\beta}, (x_i) = \gamma\hat{x_i} + \beta \tag{2}$$

Motivated by LeNet-5 convolution kernel 5×5, BN used in InceptionV3 and ReLU in Alexnet, the proposed model convolution block is built as in Figure 4.

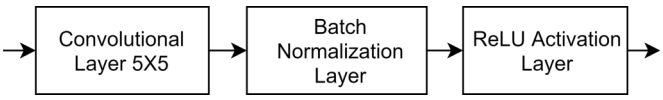


Figure 4. Proposed model convolution blocks.

The size of output feature map (FM) of each convolution block has lower size than the input feature map if no additional padding is applied. Equation (3) describes the relation between input and output FM sizes (Goodfellow et al., 2016).

$$W_y = \frac{W_x - W_k + 2P}{W_s} + 1 \tag{3}$$

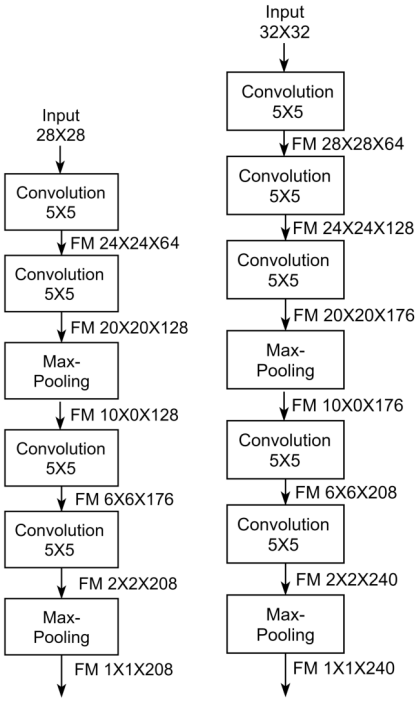
Where W_y is the width of the output, W_x is the width of the input, W_k is the width of the kernel, W_s is the width of the stride kernel and P is the number of padding pixels. For the height H, Equation (3) can be used by replacing W with H. This reduction is called the shrinkage of convolution and it limits the number of convolutional layers that the network can include (Goodfellow et al., 2016). The feature map shrinks from borders to the center as convolutional layers as added. Eventually, feature maps drop to $1 \times 1 \times channels$ (single neuron per channel) at which no more convolutional layers can be added. This is the concept of full depth used for designing the proposed architecture, Figure 5 describes the full depth idea in FDCNN, where width and height shrink by 4 according to Equation (3). In Figure 5, each feature map is shrunk to a single value and this means that the features are convoluted into a single value resulting low number of parameters and high accuracy.

The proposed FDCNN model composed basically of two stacked convolutional stages and one FC layer for 28×28 input images. Every stage has two convolution blocks and one max-pooling layer. It has a single input and a single output in all of its layers. Figure 3 shows the FDCNN architecture.

Parameter Selection

In the proposed architecture, there are some parameters have to be selected, these parameters are kernel sizes of convolution, pooling layers kerenl sizes, the number of filters (channels) in convolution layers and strides. The kernel sizes are selected to be 5×5 for convolutional layers and 2×2 for pooling layers as described in architecture design in the previous Proposed Architecture section.

In literature, the trend for selecting the number of filters is to increase the number of filters as deep as the network goes (Krizhevsky et al., 2012; Szegedy et al., 2015; Simonyan and Zisserman, 2015; He et al., 2016). Generally, the first convolutional layers learn simple features while deeper layers learn more abstract features. Selecting optimal parameters is based on heuristics or grid searches (Bengio, 2012). The rule of thumb to design a network from scratch is to start with 8-64 filters per layer and double the number of filters after each pooling layer (Simonyan and Zisserman, 2015) or after each convolutional layer (He et al., 2016). Recently, a new method was proposed to select the number of filters (Garg et al., 2018), an optimization of network structure in terms of both the number of layers and the number of filters per layer was done using principal component analysis on trained network with a single shot of



(a) 28X28 Image **(b)** 32X32 Image.

Figure 5. Full Depth concept of FDCNN.

analysis. A 9X reduction in the number of operations and up to 3X reduction in the number of parameters with less than 1% drop in accuracy is achieved upon training on the same task. In context, a modeling and optimization method (MAOM) was proposed in (Chou et al., 2019) to optimize CNN parameters by integrating uniform experimental design (UED) and multiple regression (MR), but the rule of thumb for doubling the number of filters was also applied.

One of the contributions of this research is to select the number of channels that achieves full depth. Number of filters may also be called the number of kernels, number of layer channels or layers width. The number of filters is selected to be as the same as the number of shrinking pixels in each layer from bottom to the top. Table 4 shows the shrinkage of the proposed model. From the fact that the network

Table 4. Shrinkage process in 28×28 architecture.

Layer	Shrinking Pixels	Width
Conv1	$28^2-24^2=208$	64
Conv2	$24^2-20^2=176$	128
Max-Pooling 1	—-	128
Conv3	$10^2-6^2=64$	176
Conv4	$6^2-2^2=32$	208
Max-Pooling 2	——	208

goes more deeper the following selection is made:

- The width of 4^{th} convolutional layer is 208 (the 1^{st} layer shrinkage).
- The width of 3^{rd} is 176 (the the 2^{nd} layer shrinkage).
- The max-pooling will make a loss of half in FM dimensions so the next layers shrinkage pixels will be doubled.
- The width of 2^{nd} is 128 (the double of the 3^{rd} layer shrinkage).

- The width of 1^{nd} is 64 (the double of the 4^{th} layer shrinkage).

The same parameter selection method can be applied to 32×32 input architecture as described in Table 5 to reach full depth features (single value feature) as shown in Figure 5b.

Table 5. Shrinkage process in 32×32 architecture.

Layer	Shrinking Pixels	Width
Conv1	$32^2 - 28^2 = 240$	64
Conv2	$28^2 - 24^2 = 208$	128
Conv3	$24^2 - 20^2 = 176$	176
Max-Pooling 1	—-	176
Conv4	$10^2 - 6^2 = 64$	208
Conv5	$6^2 - 2^2 = 32$	240
Max-Pooling 2	——	240

Table 6 shows the number of learnable parameters and feature memory usage for the proposed model. Memory usage is multiplied by 4 as each pixel is stored as 4-byte single float number. For 32×32 input images just another convolutional block can be added before the first convolution block in FDCNN and the width of last convolutional layer will be $32^2 - 28^2 = 240$ to get full depth of shrinkage. This layer of course affects the total number of model parameters and FM memory usage to be 2.94M and 2.51MB respectively.

Table 6. Proposed model's memory usage and learnable parameters.

Layer	Features Memory		Learnable Parameters	
Input	$28 \times 28 \times 1$	3136	0	0
Conv1	$24 \times 24 \times 64$	147456	$(5 \times 5) \times 64 + 64 =$	1664
BN+ReLU	$24 \times 24 \times 64 \times 2$	294912	$4 \times 64 =$	256
Conv2	$20 \times 20 \times 128$	204800	$5 \times 5 \times 64 \times 128 + 128 =$	204928
BN+ReLU	$20 \times 20 \times 128 \times 2$	409600	$4 \times 128 =$	512
Max-pooling1	$10 \times 10 \times 128$	51200	0	0
Conv3	$6 \times 6 \times 176$	25344	$5 \times 5 \times 128 \times 176 + 176 =$	563376
BN+ReLU	$6 \times 6 \times 176 \times 2$	50688	$4 \times 176 =$	704
Conv4	$2 \times 2 \times 208$	3328	$5 \times 5 \times 176 \times 208 + 208 =$	915408
BN+ReLU	$2 \times 2 \times 208 \times 2$	6656	$4 \times 208 =$	832
Max-pooling2	$1 \times 1 \times 208$	832	0	0
FC	$1 \times 1 \times 10$	40	$208 \times 10 + 10 =$	2090
	Total Memory	1197992 Bytes	Total Parameters	1689770

In general, DNNs give weights for all input features (neurons) to produce the output neurons, but this needs a huge number of parameters. Instead, CNNs convolve the adjacent neurons by the convolution kernel size to produce the output neurons. In the literature, the state-of-the-art architectures had high number of learnable parameters at the last FC layers. For example, VGG16 has totally 136M parameters, and after the last pooling layer the first FC layer has 102M parameters, which means more than 75% of the architecture parameters (just in one layer). AlexNet has totally 62M parameters, and the first FC layer has 37.75M parameters, which means more than 60% of the architecture parameters. In (Hirata and Takahashi, 2020), the proposed architecture has 28.68M parameters, and the first FC layer has 3.68M parameters after majority voting from ten divisions. But, by using the full depth concept to reduce FM to 1x1 size after the last pooling layer, FDCNN has just 2090 parameters from totally 1.6M parameters as seen in Table 6. The full depth concept of reducing the feature maps size to one neuron has decreased the total number of learnable parameters which make FDCNN simple and fast.

Training Process

Deep learning training algorithms were well explained in (Goodfellow et al., 2016). The proposed model is trained using stochastic gradient descent with momentum (SGDM) with custom parameters chosen after many trails, initial learning rate (LR) of 0.025, mini-batch size equals to the number of training instances divided by number of batches needed to complete one epoch, LR drop factor by half every 2 epochs, 10 epochs, 0.95 momentum and the training set is shuffled every epoch. However, those training parameters are not used for all datasets since the number of images is not constant in all of them.

After getting the first results, The model parameters are tuned by training again on ADAM with larger mini-batch size and very small LR started by 1×10^{-5}, then multiplying the batch size by 2 and LR by 1/2 every 10 epochs as long as the test error has improvement.

EXPERIMENTAL RESULTS AND DISCUSSION

All of training and testing are made on MATLAB2018 platform with GeForce 1060 (6GB shared memory GPU). The main goal of this research is to design a CNN to recognize multi-language characters of license plates but to generalize and verify the designed architecture several tests on handwritten character recognition benchmarks are done (verification process). The proposed approach showed very promising results. Table 7 summarizes the results obtained on MNIST dataset.

Table 7. Test results of FDCNN on MNIST.

Architecture	Type	Number of Layers	Error
Cireşan et al. (2010)	stacked	15	0.35%
Ciresan et al. (2011)	sparse	35	0.27%
Ciregan et al. (2012)	sparse	245	0.23%
Moradi et al. (2019)	sparse	70	0.28%
Kowsari et al. (2018)	sparse	—	0.18%
Assiri (2019)	stacked	15	0.17%
Hirata and Takahashi (2020)	sparse	28	0.16%
Byerly et al. (2020)	sparse	21	0.16%
Proposed	stacked	**12**	0.28%

It is clear that stacked CNN has not outperformed the error of 0.35% in the literature for MNIST but the approach used in (Assiri, 2019) obtained 0.17%. The proposed FDCNN performance approximately reached close to the performance of five committees CNN of (Ciresan et al., 2011). FDCNN do as the same performance as (Moradi et al., 2019) which it is a sparse design that uses Residual blocks and Inception blocks as described in the literature. However, the architecture in (Assiri, 2019) has 15 layers with 13.12M parameters while FDCNN has 12 layers with 1.69M parameters which means that FDCNN is simpler and 7 times faster (in terms of number of the parameters 13.12/1.69). The results in (Assiri, 2019) were obtained utilizing data augmentations (not used in FDCNN training), different training processes (FDCNN training process is simpler as described in the previous section) and Dropout layers before and after each pooling layer with different settings, but FDCNN has no Dropout layer and showed good results on MNIST.

On the other hand, the proposed approach is tested on MADbase, AHCD and AI9IK datasets for Arabic character recognition benchmarks to verify FDCNN and to generalize using it in Arabic ALPR systems. Table 8 describes the classification error regarding the stat-of-the-art on such datasets. The same logic of size-sensitive feature proposed in (Abdleazeem and El-Sherif, 2008) is used to solve the problem of Arabic zero character by half size reduction for Arabic zero character images (in MADbase dataset) since it has a smaller size than other characters.

As seen in Table 8, for MADbase dataset, most of the tested approaches were based on VGG architecture. Alphanumeric VGG (Mudhsh and Almodfer, 2017) reported a validation error of 0.34% that did not hold on the test set while FDCNN obtained 0.15% validation error and 0.34% test error. The proposed approach outperformed Arabic character recognition benchmarks state-of-the-arts for both digits and letters used in Arabic language with less number of layers and learnable parameters. It has succeed this verification process on these datasets too.

Table 8. Arabic character recognition benchmarks state-of-the-art and proposed approach test errors.

Dataset	Architecture	Type	Layers	Parameters	Error
MADbase 28×28	RBF SVM Abdleazeem and El-Sherif (2008)	linear	—	—	0.52%
	LeNet5 El-Sawy et al. (2017)	stacked	7	51K	12%
	Alphanumeric VGG Mudhsh and Almodfer (2017)	stacked	17	2.1M	0.34% validation
	VGG12_REGU Sousa (2018)	Average of 4 stacked CNN	66	18.56M	0.48%
	proposed	stacked	12	1.69M	**0.34%**
AHCD 32×32	CNN El-Sawy et al. (2017)	stacked	7	1.8M	5.1%
	CNN Younis (2017)	stacked	6	200K	2.4%
	VGG12_REGU Sousa (2018)	Average of 4 stacked CNN	66	18.56M	1.58%
	CNN Najadat et al. (2019)	stacked	10	Not mentioned	2.8%
	proposed	stacked	13	2.94M	**1.39%**
AI9IK 32×32	RBF SVM Torki et al. (2014)	linear	—	—	5.72%
	CNN Younis (2017)	stacked	6	200K	5.2%
	proposed	stacked	13	2.94M	**3.27%**

In Table 8, input layer is included in the determination of the number layers (Lecun et al., 1998) for all architectures and ReLU layer is not considered as a layer but BN is considered as a layer. Sousa (2018) considered convolution, pooling and FC layers when the number of layers was declared but four trained CNNs were used with softmax averaging, this is why the number of layers and learnable parameters are high. Najadat et al. (2019) did not declare the most of network parameters like kernel size in every convolution layer and they changed many parameters to enhance the model. In (Younis, 2017), 28×28 input images were used and no pooling layers were included.

On the other hand, and in the same verification process, the proposed approach is also tested on FashionMNIST benchmark to generalize using it over grayscale tiny images. As shown in Table 9, the proposed approach outperformed the stacked CNN architectures and reached near DENSER network in (Assunção et al., 2018) and EnsNet in (Hirata and Takahashi, 2020) with less layers and parameters but with a good performance. It can be said that FDCNN has a very good verification performance on FashionMNIST dataset. FDCNN outperformed (Byerly et al., 2020) results on Fashion-MNIST benchmark while (Byerly et al., 2020) outperformed FDCNN on MNIST.

Table 9. Test results of FDCNN on FashionMNIST.

Architecture	Type	Layers	Parameters	Error
SVM (Xiao et al., 2017)	linear	—	—	10.3%
DENSER (Assunção et al., 2018)	sparse	—	—	4.7%
WRN (Zhong et al., 2017)	sparse	28	36.5M	3.65%
VGG16 (Zeng et al., 2018)	sparse	16	138M	2.34%
CNN (Chou et al., 2019)	stacked	16	0.44M	8.32%
BRCNN (Byerly et al., 2020)	sparse	16	1.51M	6.34%
EnsNet (Hirata and Takahashi, 2020)	sparse	28	28.67M	4.7%
Proposed	stacked	**12**	**1.69M**	**5.00%**

All of the previous datasets were divided into training and test sets by their authors where the instances in the test set were collected from a different source (different writers for CR and different photographers for fashion) from the training set's source. The performance evaluation is done based on CNN type (stacked is simpler than spars), number of layers, number of learnable parameters and recognition error.

Furthermore, FDCNN is tested also on Arabic LP characters from KSA. However, Khaled et al. (2010) used his dataset for both training and testing, FDCNN could classify the whole dataset (as a test set) of (Khaled et al., 2010) with error of 0.46% whereas the training was done on characters collected and cropped manually from public KSA LP images. It outperformed the recognition error results of 1.78% in (Khaled et al., 2010). FDCNN has successfully verified on KSA Arabic LP characters dataset.

In this research and for more verification, FDCNN performance is also tested on both common publicly available LP benchmark characters and the new LPALIC dataset. Table 10 shows the promising results on LP benchmarks. FDCNN outperformed the state-of-the-art results on common LP datasets for isolated character recognition problem. Zemris, UCSD, Snapshots and ReId datasets were not used in the training process but the proposed FDCNN was tested on each of them as a test set to ensure that the model was fitted to character features, not to a dataset itself. For UFPR dataset, FDCNN was tested two times on UFPR test set, training on only the training set of UFPR and training on both UFPR and LPALIC characters. It is clear that FDCNN has efficiently verified on common LP benchmarks.

Table 10. Recognition error of proposed architecture on LP benchmarks datasets.

Architecture	Dataset	Layers	Parameters	Error
SVM (Panahi and Gholampour, 2017)	Zemris	—	—	3%
LCR-Alexnet (Meng et al., 2018)		12	>2.33M	2.7%
Proposed		12	1.69M	**0.979%**
OCR (Dlagnekov, 2005)	UCSD	—	—	10.5%
Proposed		12	1.69M	**1.51%**
MLP (Martinsky, 2007)	Snapshots	—	—	15%
Proposed		12	1.69M	**0.42%**
CNN (Špaňhel et al., 2017)	ReID	12	17M	3.5%
DenseNet169 (Zhu et al., 2019)		169	>15.3M	6.35%
Proposed		12	1.69M	**1.09%**
CNN (Laroca et al., 2018)	UFPR	26	43.1M	35.1%
Proposed trained just on UFPR		12	1.69M	4.29%
Proposed trained on LPALIC		12	1.69M	**2.03%**
Line Processing Algorithm (Khaled et al., 2010)	KSA	—	—	1.78%
Proposed		12	1.69M	**0.46%**
FDCNN	LPALIC	12	1.69M	0.97%

For more analysis, another test is made on the introduced LPALIC dataset to analyze the recognition error on characters per country. Table 11 describes the results. As seen in Table 11, the highest error is in classifying USA LP characters because it has more colors, drawings and shapes other than characters and also there is a small number of instances in the characters dataset. However, a very high recognition accuracy is achieved on Turkey and EU since they have the same standard and style for LPs. In Turkey, 10 digits and 23 letter is used since letters like Q, W and X are not valid in Turkish language. Additionally, FDCNN could classify Arabic LP characters with very low error. UAE characters set has a small number of cropped characters that is why it is tested just by FDCNN trained on other countries character sets.

To make robust tests, the characters were split manually and randomly as seen in Table 11. In manual split, the most difficult characters (difficult at manual labelling the character images in the dataset preparing stage) were put in the test set and the others in the training set while in random split 80% were split for training and the rest of them for testing. As described in Table 3, the number of characters per country is not equal, which resulted various recognition accuracies in Table 11. Since the number of UAE characters is not large enough to train FDCNN, Latin characters from other countries were used for training but the test was done only on UAE test set. FDCNN could learn features that give good average accuracy.

Table 11. Test recognition error per country characters with different training instances.

Characters Set	Number of Instances Train / Test	Manual Split	Trained on Other Countries	Random 80/20% Split Average Error	Random 70/10/20% Split Average Error
TR	48748 / 11755	2.67%	1.82%	0.97%	0.99%
EU	23299 / 9477	2.30%	1.07%	1.03%	0.80%
USA	5960 / 1424	10.88%	3.51%	1.96%	1.79%
UAE	1279 / 1724	—	1.51%	0.9%	1.08%
All Latin Characters	96899 / 24380	2.08%	—	0.97%	1.06%
KSA	46981 / 3018	0.43%	—	0.26%	0.30%

In the manual split in Table 11, the country's characters training and testing sets were used to train and test FDCNN. In trained on other countries, the FDCNN was trained on both the country's characters training set and other countries characters but tested only on that country's test set. In the random 80/20 split, the country's characters were split randomly into training and testing sets, and FDCNN was trained on both the split country's characters training set and other countries characters but tested only on that split country's test set, a lot of random split tests were done and the average errors were reported in the table.

Furthermore, in Table 11, validation sets were also used to guarantee in a sufficiently clear way that the results were not optimized specifically for those test sets. 70% of the dataset is randomly split for training, 10% for validation and 20% for testing. The training hyperparameters were optimized on a validation set, and the best parameters for the validation set were then be used to calculate the error on the test set. Those different test analyses were done to validate and evaluate the results and reduce the overfitting problem. In fact, the Latin characters in LPALIC have various background and foreground colors which make the classification more challenging than Arabic characters set, but FDCNN shows a promising recognition results on both and also on handwritten characters as well.

CONCLUSION

This research focused on deep learning technique of CNNs to recognize multi-language LP characters for both Latin and Arabic characters used in vehicle LPs. A new approach is proposed, analyzed and tested on Latin and Arabic CR benchmarks for both LP and handwritten characters recognition. The proposed approach consists of proposing FDCNN architecture, FDCNN parameter selection and training process. The proposed full depth and width selection ideas are very efficient in extracting features from tiny grayscale images. The complexity of FDCNN is also analyzed in terms of number of learnable parameters and feature maps memory usage.The full depth concept of reducing the feature maps size to one neuron has decreased the total number of learnable parameters while achieving very good results. Implementation of FDCNN approach is simple and can be used in real time applications worked on small devices like mobiles, tablets and some embedded systems. Very promising results were achieved on some common benchmarks like MNIST, FashionMNIST, MADbase, AIA9K, AHCD, Zemris, ReId, UFPR and the newly introduced LPALIC dataset. FDCNN performance is verified and compared to the state-of-the-art results in the literature. A new real LPs cropped characters dataset is also introduced. It is the largest dataset for LP characters in Turkey and KSA. More tests can be done on FDCNN for future work to be the core of CNN processor. Also, more experiments can be conducted to hybrid FDCNN with some common blocks like residual and inception blocks. Additionally, the proposed full depth approach may be applied to other stacked CNNs like Alexnet and VGG networks.

REFERENCES

Abdleazeem, S. and El-Sherif, E. (2008). Arabic handwritten digit recognition. *International Journal of Document Analysis and Recognition (IJDAR)*, 11(3):127–141.

Asif, M. R., Chun, Q., Hussain, S., Fareed, M. S., and Khan, S. (2017). Multinational vehicle license plate detection in complex backgrounds. *Journal of Visual Communication and Image Representation*, 46:176 – 186.

Asif, M. R., Qi, C., Wang, T., Fareed, M. S., and Raza, S. A. (2019). License plate detection for multi-national vehicles: An illumination invariant approach in multi-lane environment. *Computers & Electrical Engineering*, 78:132 – 147.

Assiri, Y. S. (2019). Stochastic optimization of plain convolutional neural networks with simple methods. In *15th International Conference on Machine Learning and Data Mining, MLDM*, volume 2, pages 833–844. ibai-publishing.

Assunção, F., Lourenço, N., Machado, P., and Ribeiro, B. (2018). DENSER: deep evolutionary network structured representation. *CoRR*, abs/1801.01563.

Bengio, Y. (2012). *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bulan, O., Kozitsky, V., and Burry, A. (2015). Towards annotation free license plate recognition. In *IEEE 18th International Conference on Intelligent Transportation Systems*, pages 1495–1499.

Bulan, O., Kozitsky, V., Ramesh, P., and Shreve, M. (2017). Segmentation- and annotation-free license plate recognition with deep localization and failure identification. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2351–2363.

Byerly, A., Kalganova, T., and Dear, I. (2020). A branching and merging convolutional network with homogeneous filter capsules. *ArXiv*, abs/2001.09136.

Chou, F., Tsai, Y., Chen, Y., Tsai, J., and Kuo, C. (2019). Optimizing parameters of multi-layer convolutional neural network by modeling and optimization method. *IEEE Access*, 7:68316–68330.

Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220.

Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649.

Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *International Conference on Document Analysis and Recognition*, pages 1135–1139.

Comelli, P., Ferragina, P., Granieri, M. N., and Stabile, F. (1995). Optical recognition of motor vehicle license plates. *IEEE Transactions on Vehicular Technology*, 44(4):790–799.

Dlagnekov, L. (2005). Video-based car surveillance: License plate, make, and model recognition. Master's thesis, University of California, San Diego.

Dorbe, N., Jaundalders, A., Kadikis, R., and Nesenbergs, K. (2018). Fcn and lstm based computer vision system for recognition of vehicle type, license plate number, and registration country. *Automatic Control and Computer Sciences*, 52(2):146–154.

Du, S., Ibrahim, M., Shehata, M., and Badawy, W. (2013). Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325.

El-Sawy, A., EL-Bakry, H., and Loey, M. (2017). Cnn for handwritten arabic digits recognition based on lenet-5. In Hassanien, A. E., Shaalan, K., Gaber, T., Azar, A. T., and Tolba, M. F., editors, *International Conference on Advanced Intelligent Systems and Informatics*, pages 566–575. Springer International Publishing.

El-Sawy, A., Loey, M., and EL-Bakry, H. (2017). Arabic handwritten characters recognition using convolutional neural network. *WSEAS Transactions on Computer Reseach*, 5:11–19.

El-Sherif, E. A. and Abdelazeem, S. (2007). A two-stage system for arabic handwritten digit recognition tested on a new large database. In *Artificial Intelligence and Pattern Recognition*.

Eltay, M., Zidouri, A., and Ahmad, I. (2020). Exploring deep learning approaches to recognize handwritten arabic texts. *IEEE Access*, 8:89882–89898.

Fernández, F., Negri, P., Mejail, M., and Jacobo, J. (2011). A multi-style license plate recognition system based on tree of shapes for character segmentation. In San Martin, C. and Kim, S.-W., editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 443–450. Springer Berlin Heidelberg.

Garg, I., Panda, P., and Roy, K. (2018). A low effort approach to structured CNN design using PCA. *CoRR*, abs/1812.06224.

Gonçalves, G. R., Menotti, D., and Schwartz, W. R. (2016). License plate recognition based on temporal redundancy. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2577–2582.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Guha, R., Das, N., Kundu, M., Nasipuri, M., and Santosh, K. C. (2020). Devnet: An efficient cnn architecture for handwritten devanagari character recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 34(12):2052009.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Henry, C., Ahn, S. Y., and Lee, S. (2020). Multinational license plate recognition using generalized character sequence detection. *IEEE Access*, 8:35185–35199.

Hirata, D. and Takahashi, N. (2020). Ensemble learning in cnn augmented with fully connected subnetworks. *ArXiv*, abs/2003.08562.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

Huang, G., Liu, Z., v. d. Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 1mb model size. *CoRR*, abs/1602.07360.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*, volume 37 of *ICML'15*, pages 448–456. JMLR.org.

Khaled, M. A., Rached, N. Z., and Hasan, R. O. (2010). Pixel density: Recognizing characters in saudi license plates. In *10th International Conference on Intelligent Systems Design and Applications*, pages 308–313.

Kowsari, K., Heidarysafa, M., Brown, D. E., Meimandi, K. J., and Barnes, L. E. (2018). Rmdl: Random multimodel deep learning for classification. In *Proceedings of the 2nd International Conference on Information System and Data Mining*, ICISDM '18, page 19–28, New York, NY, USA. Association for Computing Machinery.

Kraupner, K. (2003). Using multilayer perceptron to recognize numeric-alphanumeric characters on license plates. Bsc. eng. thesis, University of Zagreb, Croatia.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R., and Menotti, D. (2018). A robust real-time automatic license plate recognition based on the YOLO detector. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–10.

Le, N. Q. K. and Nguyen, V.-N. (2019). Snare-cnn: a 2d convolutional neural network architecture to identify snare proteins from high-throughput sequencing data. *PeerJ Computer Science*, 5:e177.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, H., Wang, P., and Shen, C. (2019). Toward end-to-end car license plate detection and recognition with deep neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1126–1136.

Martinsky, O. (2007). Algorithmic and mathematical principles of automatic number plate recognition systems. B.sc. eng. thesis, Brno University of Technology, Croatia.

Meng, A., Yang, W., Xu, Z., Huang, H., Huang, L., and Ying, C. (2018). A robust and efficient method for license plate recognition. In *24th International Conference on Pattern Recognition (ICPR)*, pages 1713–1718.

Moradi, R., Berangi, R., and Minaei, B. (2019). Orthomaps: an efficient convolutional neural network with orthogonal feature maps for tiny image classification. *IET Image Processing*, 13(12):2067–2076.

Mudhsh, M. and Almodfer, R. (2017). Arabic handwritten alphanumeric character recognition using very deep neural network. *Information*, 8(3):105.

Najadat, H. M., Alshboul, A. A., and Alabed, A. F. (2019). Arabic handwritten characters recognition using convolutional neural network. In *IEEE 10th International Conference on Information and Communication Systems (ICICS)*, pages 147–151.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised*

Feature Learning 2011.

Panahi, R. and Gholampour, I. (2017). Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):767–779.

Park, S., Yoon, H., and Park, S. (2019). Multi-style license plate recognition system using k-nearest neighbors. *KSII Transactions on Internet and Information Systems*, 13(5):2509–2528.

Salemdeeb, M. and Erturk, S. (2020). Multi-national and multi-language license plate detection using convolutional neural networks. *Engineering, Technology Applied Science Research*, 10(4):5979–5985.

Selmi, Z., Halima, M. B., Pal, U., and Alimi, M. A. (2020). Delp-dar system for license plate detection and recognition. *Pattern Recognition Letters*, 129:213 – 223.

Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung, and Sei-Wan Chen (2004). Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems*, 5(1):42–53.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Sousa, I. P. D. (2018). Convolutional ensembles for arabic handwritten character and digit recognition. *PeerJ Computer Science*, 4:e167V.

Špaňhel, J., Sochor, J., Juránek, R., Herout, A., Maršík, L., and Zemčík, P. (2017). Holistic recognition of low quality license plates by cnn using track annotated data. In *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Torki, M., Hussein, M. E., Elsallamy, A., Fayyaz, M., and Yaser, S. (2014). Window-based descriptors for arabic handwritten alphabet recognition: A comparative study on a novel dataset. *ArXiv*, abs/1411.3519.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xie, L., Ahmad, T., Jin, L., Liu, Y., and Zhang, S. (2018). A new cnn-based method for multi-directional car license plate detection. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):507–517.

Xu, Z., Yang, W., Meng, A., Lu, N., and Huang, H. (2018). Towards end-to-end license plate detection and recognition: A large dataset and baseline. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 255–271.

Younis, K. (2017). Arabic handwritten character recognition based on deep convolutional neural networks. *Jordanian Journal OF Computers and Information Technology (JJCIT)*, 3(3):186–200.

Yépez, J., Castro-Zunti, R. D., and Ko, S. (2019). Deep learning-based embedded license plate localisation system. *IET Intelligent Transport Systems*, 13(10):1569–1578.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press.

Zeng, S., Zhang, B., Zhang, Y., and Gou, J. (2018). Collaboratively weighting deep and classic representation via l_2 regularization for image classification. In *Proceedings of The 10th Asian Conference on Machine Learning*, volume 95, pages 502–517.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2017). Random erasing data augmentation. *CoRR*, abs/1708.04896.

Zhu, L., Wang, S., Li, C., and Yang, Z. (2019). License plate recognition in urban road based on vehicle tracking and result integration. *Journal of Intelligent Systems*, 0(0).