Food Recognition

Igor Tica, Filip Basara
Chair of Informatics, Faculty of Technical Sciences Novi Sad, Serbia

Abstract*—Image recognition is attracting a lot of attention lately, because of tremendous progress that has been made in this area. The main credit for this success belongs to the Convolutional Neural Network (CNN) – a kind of model that shows the best performance on hard visual recognition tasks. In this paper, we discuss a more specific problem – food recognition. Since CNNs are considered to be state of the art for image recognition problem, our approach is based on them. Firstly, we explain basic theory behind CNNs, needed to understand the specific architectures used for food recognition. Secondly, we reference two successful architectures CNN – Inception-ResNet and WISeR, that have respectively achieved 90.14% and 90.27% precision at their peak on the Food-101 dataset. Accordingly, we explain that successful models trained for general image recognition problems can be used as a starting point and retrained using a different dataset in order to tackle a specific problem such as food recognition on images. Also, in this paper we present our simple CNN architecture, designed to classify 10 types of foods and implemented in the Keras framework. We discuss our selection of hyper-parameters, activation functions and, optimization methods. Our architecture reached peak precision of 76% after 17 epochs. All solutions are trained on a Food-101 dataset, that has 101 000 images. The main limitation when training our network was the lack of technical resources, that didn't allow us to train more complex CNN architectures and recognize more types of food. In future work, we will attempt to adapt the Inception-ResNet architecture for this specific problem and implement it within a mobile device application.

Keywords*—computer vision; neural network; deep learning; food recognition

I. INTRODUCTION

Food recognition has become very popular lately, because of great advancements in the field of computer vision and deep neural networks. Furthermore, with increased image sharing on the Internet and social networks, there is a need to exploit those images to create applications that will simplify everyday life. That being said, food recognition can have a substantial impact in health care and Internet of Things. It could be integrated in software of smart fridges, calorie counting and dietary assessment applications where it would be highly beneficial to tackle the rapid increase of diseases related to excessive or wrong food intake.

The power of computer vision and pattern recognition is still far from matching human capabilities, especially when it comes to classifying an image whose features from the same class might have more differences than features from another classes. This is just the case with food recognition where a particular food dish may be prepared in thousands of different ways, yet it is essentially the same food.

Variations within one class (Image 1) is a strong source of uncertainty, since the recipe for the same food can vary depending on the location, the available ingredients and, last but not least, the personal taste of the cook. On the other hand, different foods may look very similar (e.g., soups where the main ingredients are hidden below the liquid level), thus inter-class confusion is a source of potential problems too. Despite such issues, a quick, partial view, of a food image is often sufficient for a human to recognize the food dish. This remarkable ability inevitably tells us that food images have distinctive properties that made task tractable, regardless of the non-trivial challenges.

Image 1 - The food recognition problem is characterized by large intra class variations.

There are various approaches in solving food recognition problem that are based on deep learning algorithms. They use very different principles, where some older approaches used simple architectures with several layers that are added sequentially (i.e. AlexNet [15]), contrary to current ones, that consist of modules with a number of layers, and significantly more connections between them (i.e. Inception model [18]).

In this paper, we discuss about two successful architectures – Inception-ResNet [10] and WISeR [4], that have respectively achieved 90.14% and 90.27% precision at their peak in the food recognition problem. Their approach is based on the Convolutional Neural Network (CNN), which is state of the art in image recognition [15]. We also discuss our simple CNN architecture, that solves the problem of classifying food images to 10 different classes: dumplings, edamame, French fries, red velvet cake, spaghetti carbonara, waffles, lobster bisque, mussels, onion rings and pizza. Our architecture, trained on a Food-101 [19] dataset, reached peak precision of 76% after 17 epochs.

The overall organization of our paper is as follows. Section II introduces related work on this field. In section III, we analyze the most successful architectures and their performances in the food recognition problem. Section IV describes the dataset details, while in section V we discuss the evaluation results of our proposed algorithms. We make concluding remarks in section VI.

II. Related work

Inspired by the advances of deep learning technique, some researchers have applied deep learning for visual-based food image recognition. The majority of these techniques for food recognition are based on traditional signal processing techniques with hand-engineered features. Recently, due to the occurrence of large annotated dataset like ImageNet [13], Microsoft COCO [14], and the development of powerful machines equipped with high quality Graphics Processing Units (GPUs), it is plausible to train large and complex CNN models for accurate recognition, which surpassed most of the methods adopted using hand-crafted features [15].

Yang et al. [7] proposed a method to recognize fast food using the relative spatial relationships of local features of the ingredients followed by a feature fusion method. This method only works for a small number of food categories (61 foods) and is difficult to extend to composite or homemade food. Matsuda et al. proposed an approach for multiple food recognition using a manifold ranking-based approach and co-occurrence statistics between food items, which were combined to address the recognition of multiple food items on a single image [24]. However, this type of solution is computationally intensive and may not be practical for deployment within the mobile cloud-computing platform or as an application. A sequence of papers from a Purdue University TADA project [8] covered food item identification, food volume estimation, as well as other aspects of dietary assessment, such as mobile interface design and food image database development.

In a paper by Kawano et al [12], the researchers developed an Android application to collect and label food images. They also created a food image database named UEC-256 food image dataset. With this dataset, they first conducted some experiments using Scale-invariant feature transform (SIFT) [28] features and Support Vector Machine (SVM) [30], which showed much better results compared with Radio Frequency Identification (RFID) [29]. Then, they applied the famous AlexNet CNN [15] on the same datasets and it outperformed the SIFT-SVM based method.

III. Deep Learning approach

Neural network (NN) is a learning model comprised of nodes and their connections, inspired by biological neural networks. It consists of an input, a single hidden and an output layer, that are fully connected. A layer is a set of nodes, also known as neurons. Deep Neural Network (DNN) is a neural network with multiple hidden layers between input and output layers. There are different variations of DNNs, such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN) etc.

Deep learning aims to learn multiple levels of representation and abstraction that help infer knowledge from data such as images, videos, audio, and text. It is making astonishing gains in computer vision, speech recognition, multimedia analysis, and drug designing. Briefly speaking, there are three main classes of deep learning techniques: purely supervised learning algorithms (e.g., Deep Convolutional Network), unsupervised and semi-supervised learning algorithms (e.g., Denoising Autoencoders [16], Restricted Boltzmann Machines [18], and Deep Boltzmann Machines [17]). Our proposed approach belongs to the first category (supervised learning algorithms).

Currently, CNNs are the state of the art in computer vision and in further research our focus will be on them. With the help of large-scale and well-annotated dataset like ImageNet, it's now feasible to perform large scale supervised learning using Convolutional Neural Network (CNN).

A. Convolutional Neural Network (CNN)

Convolutional Neural Network is a class of DNN, used primarily in computer vision. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were manually engineered.

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume. In a traditional neural net, each element of the weight matrix is used exactly once when computing the output of a layer. Instead, we will connect each neuron only to a local region of the input volume (Image 2). This feature of a CNN is called local connectivity [25]. The spatial extent of this connectivity is a hyper-parameter called the receptive field of the neuron (equivalently this is the filter size) (Image 2). The extent of the connectivity along the depth axis is always equal to the depth of the input volume.

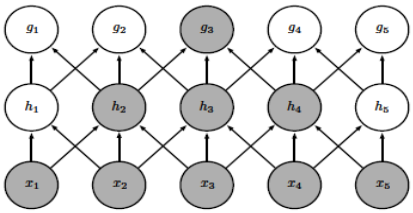

Image 2 - Sparse or local connectivity, viewed from above. We highlight one output unit, g3, and highlight the input units in h that affect this unit. These units are known as the receptive field of g3. The receptive field of the units in the deeper layers of a convolutional network is larger than the receptive field of the units in the shallow layers. This means that even though direct connections in a convolutional net are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image.

A parameter sharing scheme is used in convolutional layers to control the number of free parameters. It relies on one reasonable assumption: if a feature is useful to compute at some spatial position, than it should also be useful to compute at other positions. In other words, denoting a single 2-dimensional slice of depth as a depth slice (e.g. a volume of size [55x55x96] has 96 depth slices, each of size [55x55]), we constrain the neurons in each depth slice to use the same weights and bias[1]. Since all neurons in a single depth slice share the same parameters, then the forward pass in each depth slice of the convolutional layer can be computed as a convolution of the neuron's weights with the input volume.

[1] Bias is a constant which helps the model to deal with non-linearity in a way that it can fit best for the given data.

Besides input and output layers, a CNN can consist of convolutional, activation, pooling, fully connected layers. In following paragraphs we will explain each layer individually.

The convolutional layer [25] is the core building block of a CNN. Its parameters consist of a set of learnable filters or kernels. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. Therefore, it is common to refer to the sets of weights as a filter (or a kernel), which is convolved with the input. During the forward pass, we slide (convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position (Image 3). The result is a 2-dimensional activation map that gives the responses of that filter at every spatial position to produce the output volume.

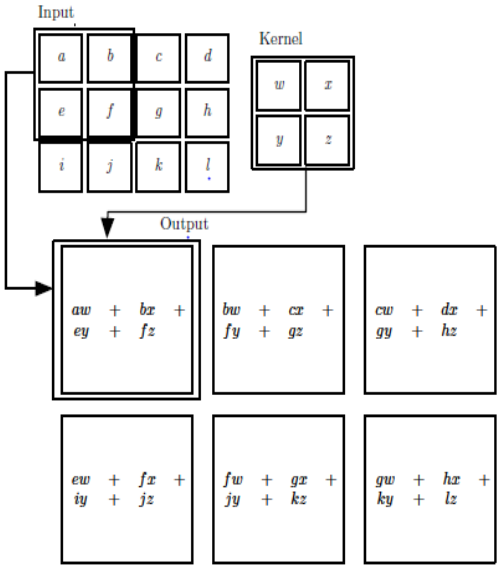

Image 3 - An example of 2-D convolution. We restrict the output to only positions where the kernel lies entirely within the image, called "valid" convolution in some contexts. We draw boxes with arrows to indicate how the upper-left element of the output tensor (matrix) is formed by applying the kernel to the corresponding upper-left region of the input tensor.

Pooling layers [25] are commonly inserted in-between successive convolutional layers in a CNN architecture. Its function is to progressively reduce the spatial size of the representation (Image 4) to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. Other popular pooling function is the average pooling, which uses the average value from each of a cluster of neurons at the prior layer. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down-samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation[2].

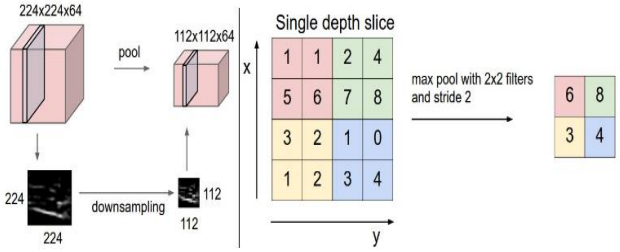

Image 4 - Pooling layer down-samples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common down-sampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (2x2 square to the most right).

The activation function of a node defines how the output of that node is calculated based on given set of inputs. Popular activation functions are Rectified linear unit (ReLU), Sigmoid, TanH etc [31]. ReLU applies an elementwise activation function, such as the max(0,x) thresholding at zero and leaves the size of the volume unchanged .

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer neural network. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

In order to make the output more compatible and intuitive, a fully connected layer has an activation function, most common one being the Softmax function[3]:

$$P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \quad (1)$$

The Softmax function assists the fully connected layer in classifying the output by normalizing the input to its probabilistic interpretation. More precisely, it takes a vector of activation values and normalizes it to a vector of values between zero and one. Each number in the resulting vector represents the probability that a specific feature is a desired output.

The successful classification algorithm has two major components: a score function that maps the raw data (e.g. image pixels) to class scores, and a loss function that quantifies the agreement between the predicted scores and the ground truth labels. This can then be observed as an optimization problem in which the loss function is minimized with respect to the parameters of the score function. Loss

[2] Max pooling uses the maximum value from each of a cluster of neurons at the prior layer.

[3] Softmax formula assigns the normalized probability to the correct label y_i given the image x_i and parameterized by W. Softmax classifier interprets the scores inside the output vector f as unnormalized log probabilities.

functions are used to measure our unhappiness with predicted outcomes. Intuitively, the loss will be high if we're doing a poor job of classifying the training data, and it will be low if we're doing well.

B. WISeR Architecture

In various scientific researches, some architectures were found to be more useful than others. The quest for optimal architecture that solves one specific problem often consists of trying different layouts of layers and adjusting the hyper-parameters. In the following paragraphs, we describe WISeR architecture [4]. This model shows great performance and it achieves better results than current state of the art apporaches [4].

The WISeR model consists of a single deep network with two main branches: a residual network branch, and a slice network branch with a slice convolutional layer. The residual network encodes generic visual representations of food images. The slice network specifically captures the vertical food layers. Information gathered from these two branches are then used and processed in the fully connected layers that emit a classification prediction.

The authors recognized the potential in residual learning [22] (Image 5) and they made a great effort to use its effectiveness of learning the parameters of very deep neural networks (e.g., with more than 20 layers). The starting point in their idea is the assumption that with a given input x, a shallow network with few stacked non-linear layers can approximate a mapping function M(x)[4]. Furthermore, it then implies that network with the same structure can approximate the residual function F(x) = M(x) – x [5]. Both learning the approximation of the mapping function M(x) or the residual function F(x) [22] is feasible, but the ease of such a process is significantly different. Indeed, deep networks trained to approximate the mapping function M(x) suffer from a degradation that does not appear in networks trained on approximating the residual function F(x) [23]. This explains the effectiveness of residual learning for very deep networks.

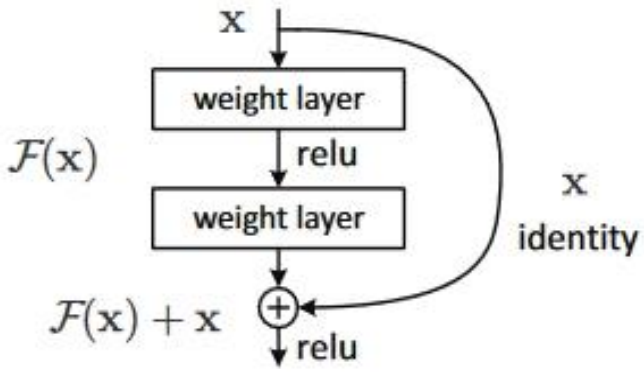

Image 5 - Residual learning: a building block.

[4] mapping function attempts to classify the given image into specified class labels, sub-segments, captions, or whatever the output is

[5] we assume that the input and output have the same dimensionality

Another network that authors introduced is slice network. They used the modified concept of convolution in it, named slice convolution (Image 6). This approach did better on extracting relevant image feature, because it considered the vertical traits of food dishes. That is because of the fact it learned the parameters of a convolution kernel that had the same width as the input image. In such a way, it acted as a vertical layer feature detector. Moreover, they applied slice pooling, more specifically max pooling on vertically elongated windows, because it is not guaranteed that the vertical layers appears in the same position and the output of the slice convolution might be different depending on the location of such vertical traits.

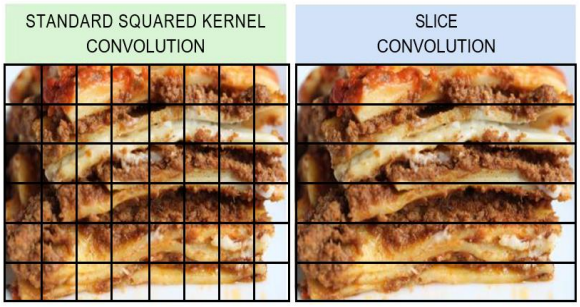

Image 6 - On the left picture there is standard squared convolutional kernel. It is commonly used in deep learning architectures for food recognition. On the right picture, there is proposed slice convolutional kernel that captures the vertical layer structure of some food dishes.

C. Our approach

The main CNN architecture we used for this problem consists of 2 convolutional layers and a single fully connected layer, a Softmax activation function and the Adagrad optimizer [27]. Pooling and ReLU activation was applied on each convolutional layer, where the output from pooling is two times smaller than the input. To overcome overfitting, we used dropout [1], batch normalization [3] and L2 regularization [11]. Dropout was the most successful method [1], while other methods performed similarly. We also combined dropout and L2 regularization, which didn't result in some significant improvements. Adagrad [26] was proven to be the best optimizing function for our problem, while stochastic gradient descent and Adam showed similar results.

The choice of adequate hyper-parameters for this architecture was mostly experimental. We trained models with different hyper-parameters on the same dataset to see which performed better. Filter size for the first convolutional layer is 32, which was found to be optimal with the 64x64x3 size of the input. Size of the kernel is 5x5, while the stride is 1x1. The algorithm used for the first pooling layer is max pooling, with a map of 2x2 and a stride of 2x2. A stride is set to make sure every pixel in the original image or feature map is covered and generates the corresponding output in the output feature map. To reduce overfitting, the first convolutional layer uses the L2 regulizer and the dropout with 0.5 chance of resetting a neuron. Before proceeding to the fully connected layer, the input is flattened (converted to a one-dimensional matrix). The

fully connected layer has 128x128 neurons and a ReLU activation function. It also has a dropout with 0.5 probability of resetting a neuron. Lastly, the architecture has a classification layer, with a Softmax activation function and 10 neurons, each representing one type of food. This model uses an Adagrad optimizer, and cross-entropy, for calculating loss. Cross-entropy measures the average number of bits needed to identify an event drawn from the set.

We implemented our solution for food recognition problem using Keras, an open source neural network library written in Python. Keras is designed to enable fast experimentation with deep neural networks and it focuses on being user-friendly, modular, and extensible. Furthermore, it contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. These are the main reasons we decided to use this specific library in development of our solution.

Models in Keras can come in two forms – Sequential and via the Functional API. We chose to use Sequential, because of the fact it allows us to easily stack sequential layers (and even recurrent layers) of the network in order from input to output.

D. Pre-trained models and the history of CNN

Pre-trained models are deep learning models with pre-trained weights. A pre-trained model can be used as a starting point to learn a new task. These models can be used for prediction, feature extraction and fine-tuning. A problem with pre-trained models is that they are trained for a specific problem, and often can't replicate the same results when tasked with something different. Since modern CNNs take 2-3 weeks to train across multiple GPUs on ImageNet, it is common to see people release their final CNN checkpoints for the benefit of others who can use the networks for fine-tuning.

Subsequent theoretical proof and experimental results both show that large scale pre-trained models in large domain, with specific small scale unlabeled data in another domain, will give excellent result in image recognition and object detection [20]. To address the issue of limited abilities of feature representation, many researchers have proposed more complex CNN network structure. On the other hand, ReLU is also proposed to make it converge faster and also gains a better accuracy. Most of current researchers have put efforts in making the network deeper and avoiding the overfitting problem.

CNN become increasingly powerful in large scale image recognition after Krizhevsky et al. won the first prize in ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 [21] with the introduction of AlexNet [15]. AlexNet (Image 7) has a simple architecture with 60 million parameters and 650,000 neurons and consists of five convolutional layers. Those layers are followed by max-pooling layers, and three globally-connected layers with a final 1000-way Softmax layer.

AlexNet was the coming out party for CNNs in the computer vision community. This was the first time a model

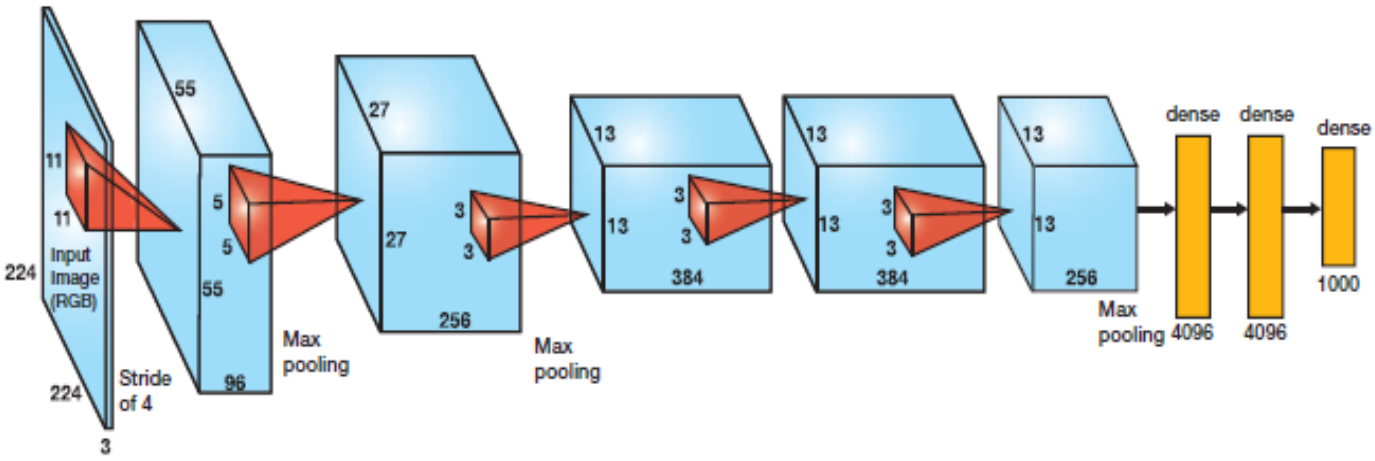

Image 7 – The AlexNet architecture

performed so well on a historically difficult ImageNet dataset. Utilizing techniques that are still used today, such as data augmentation and dropout, this paper really illustrated the benefits of CNNs and backed them up with record breaking performance at the 2012 ILSVRC.

After that, there are several symbolic milestones in the history of CNN development, which are ZFNet by Zeiler and Fergus[9], VGGNet by Simonyan et al. [5] (VGG stemps from the Visual Geometry Group at the Oxford Engineering Science Department), GoogLeNet (Inception-v1) [6],[10] and Inception-ResNet's (Residual Networks) [10] by Szegedy et al. Inception-ResNet is a 22 layer CNN and this was one of the first CNN architectures that strayed from the general approach of simply stacking convolutional and pooling layers on top of each other in a sequential structure.

The most successful general architecture and state of the art for image recognition is the Inception-ResNet model,

designed in 2016. It was proven to be superior on multiple challenging datasets, usually performing with accuracy of over 80%. When fully trained on the Food-101 dataset [19], it's precision ranges from 80% in individual projects to 90.14% at a food recognition contests.

IV. Dataset

Since deep learning-based algorithms require large datasets, we decided to use the challenging Food-101 dataset [19].

Image 8 - Each column is a different type of food from a Food-101 dataset.

For our purposes, we selected ten classes. These images were chosen from the Food-101 dataset, because we considered them to have the most distinct and representative colors and patterns.

This dataset consists of 101 categories and each category has 1000 images, in total 101,000 images (Image 8). However, since all these data were collected by food sharing websites, images do not contain any information indicating the food location. Each image only contains the label information indicating the food type. Most of the images are popular western food images. Images haven't been taken in laboratory conditions, but are randomly taken pictures by people around the globe.

B. Image Preprocessing for CNN

Even though CNNs are optimized to require minimal image preprocessing, we decided to improve image quality by applying the histogram equalization [2]. This method helps our architecture to better learn patterns between images from the same class.

We applied Histogram Equalization algorithm to increase the contrast and luminance. Image preprocessing effectively handles the problem when the pictures were taken in different environment background, which speeds up the learning pace and slightly improves the output accuracy. In future work, we are planning on exploring more image preprocessing and image augmentation techniques.

V. Results and Dicscussions

In this section, we will discuss about the performance and architectures of our model. To analyze the performance, we will use the output from the model fit function provided by the Keras framework. We will analyze the training accuracy and the validation (testing) accuracy of our model. Dataset was split to 80% for training and 20% for validation/testing. Training and validation were done for 20 epochs, with the batch size of 20. Adagrad was chosen as the optimizer. We didn't experiment with activation functions, since ReLU is a standard for CNNs. In this section, we will discuss about how we tweaked those parameters in attempt to achieve an optimal model.

When training the CNN, we tweaked hyper-parameters, to decide which would bring the biggest improvements in validation accuracy and validation loss. First of all, we fed the network with different image sizes, first 64x64, and then 128x128. The idea was that the better image quality would improve network's ability to learn weights. It didn't result in any improvements, since the validation accuracy of the bigger image size was mostly the same, if not lower. Another downside of the bigger image size was the time needed to complete a single epoch – it took a 128x128 image size approximately four times more time to complete an epoch, than it took for a 64x64 image size. That is proportional with the number of parameters of both versions – bigger image size had four times more parameters. Also, it is worth noting that the bigger image size took more epochs to reach its peak than the smaller one. That being taken into consideration, it is safe to say that the smaller image size is more effective and that increasing the image size doesn't improve the performance of the network.

Secondly, we tweaked the number of filters in the convolutional layer and increased it to 64, compared to the original size of 32. Results were similar to the ones where we increased the image size. It resulted in approximately 1% lower validation accuracy, but took twice the time. The training accuracy and loss increased a lot faster compared to the smaller volume size, which resulted in over-fitting problems. The reason for this could be the simplicity of our network.

We also tweaked the size of the fully connected layer, increasing it to 192x192 and 256x256, compared to the original 128x128. The 128x128 had the best performance, taking the least amount of time and having the best validation accuracy and loss.

Finally, we experimented with different optimization methods. First optimization method we used was the standard Stochastic Gradient Descent (SGD), which was followed by Adam [27] and Adagrad [26]. Performance rates were as following, starting from the worst: SGD, Adam, Adagrad. In general, Adagrad's validation accuracy was 3% bigger than SGD's and 1% bigger than Adam's.

One of our main problems was over-fitting. It was partially repressed by including dropout and regularization techniques, but still presented the biggest problem in increasing the validation accuracy. Dropout proved to be the more effective method. We applied it twice, both times with the probability of 0.5. Increasing the probability didn't result in improvements, but decreasing it lowered the validation accuracy of the model. Over-fitting problem was obvious since the training accuracy increased rapidly and consistently throughout each epoch, while the validation accuracy starts to stagnate after a few epochs. That posed a problem since it didn't allow the model to learn new patterns. Even better indicator on this issue was the validation loss, which decreased after a first few epochs, and

then started to stagnate, while the training decreased steadily and seemed to be converging.

VI. Conclusion and Future Work

In this paper, we proposed a simple CNN model, trained on a Food-101 dataset to classify 10 types of food. We experimented with different architectures by tweaking hyper-parameters, adding or removing layers, using different optimization algorithms, using image preprocessing, etc. The result was the architecture capable of classifying 10 types of food with a precision of 76%. The main reason that the better precision couldn't be achieved in this project is the overfitting problem, which is probably caused by the complexity of the dataset and the simplicity of our model.

Our main goal in this paper was to create and evaluate an architecture that would not require expensive hardware and cloud computing services, but would still be effective in classifying food images. This architecture is supposed to offer us an insight to better understanding CNNs and to give us some practical knowledge for future projects. We deliberately chose a challenging dataset, that required image preprocessing to make the images clearer for the CNN.

Our aspiration was to learn as much as possible about successful CNN architectures, which we plan to use on our future projects. We would single out two models - one for being the state of the art in image recognition, and the other one for being the best architecture for specifically food recognition. The first one is Inception-ResNet, which achieved 90.14% on the Food-101 dataset. As for key factors for improving classification accuracy, they mentioned random data augmentation by scaling, cropping and dense evaluation. The second one is the WISeR architecture, that employed image recognition techniques special for food. The authors introduced a new deep scheme that was designed to handle the food structure. Specifically, they used concepts of residual deep network and slice convolution block to capture the vertical food layers. They showed that this approach improves accuracy of recognizing food and achieves better performances than existing approaches. In future, our plan is to use cloud services to train an Inception-ResNet model on this dataset with more classes and higher precision.

References

List of references:

[1] Srivastava, N., Hinton, G., Krizhevsky, A., Salakhutdinov, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Article in Journal of Machine Learning Research, June 2014 – nice and useful work on dropout technique, that we used in our solution

[2] "Histograms - 2: Histogram Equalization", OpenCV documentation, URL: http://docs.opencv.org/3.1.0/d5/daf/tutorial/py_histogram_equalization.html - method that we used in our work.

[3] Ioffe, S., Szegedy, C., " Batch normalization: Accelerating deep network training by reducing internal covariate shift". In Proceedings of ICML, 2015 – a paper on batch normalization

[4] Martinel, N., Foresti, G. L., Micheloni, C., "Wide-Slice Residual Networks for Food Recognition", published in ArXiv, URL: https://arxiv.org/pdf/1612.06543.pdf , 2016 – very important paper, real breakthrough – their solution is state of the art for food recognition; we analized their arhictecture

[5] Simonyan, K, Zisserman, A., "Very Deep Convolutional Networks for Large-Scale Image Recognition", published in ArXiv, URL: https://arxiv.org/pdf/1409.1556.pdf , April 2015 – paper about one of the top models in the 2014 ImageNet challenge

[6] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., "Rethinking the Inception Architecture for Computer Vision", published in ArXiv, URL: https://arxiv.org/pdf/1512.00567.pdf , December 2015 – we found some useful information about Incepition-v2, which we also mention in out paper

[7] Yang, S., Chen, M., Pomerleau, D., Sukthankar, R., "Food Recognition Using Statistics of Pairwise Local Features", 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010 –

[8] TADA: Technology Assisted Dietary Assessment at Purdue University, West Lafayettem Indiana, USA, http://www.tadaproject.org/

[9] Fergus, R., Zeiler, D. M., "Visualizing and Understanding Convolutional Networks", published in ArXiv, URL: https://arxiv.org/pdf/1311.2901.pdf , 2013 – paper about one of the first AlexNet descendants

[10] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A., "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", published in ArXiv, URL: https://arxiv.org/pdf/1602.07261.pdf , August 2016 – paper on an impart in image recongition by Inception-ResNet and Inception v4

[11] Cortes, C., Mohri, M., Rostamizadeh, A., " L2 regularization for learning kernels", In Uncertainty in Artificial Intelligence, 2009 – a paper on L2 regularization

[12] Kawano, Y., Yanai, K., "FoodCam: A real-time food recognition system on a smartphone", Multimedia Tools and Applications Journal, 2015 – one of the first mobile food recogniton applications

[13] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., "ImageNet: A Large-Scale Hierarchical Image Database", IEEE Computer Vision and Pattern Recognition (CVPR), 2009 – human-anotated image database

[14] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C. L., "Microsoft COCO: Common objects in ´ context", published in ArXiv, URL: https://arxiv.org/abs/1405.0312 , 2014 - a large-scale object detection, segmentation, and captioning dataset

[15] Krizhevsky, A., Sutskever, I., and Hinton, G., "ImageNet classification with deep convolutional neural networks", In NIPS, 2012 – a paper about one of the most influential CNNs to date

[16] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. A, "Extracting and composing robust features with denoising autoencoders", 2008 – a paper introducing Denoising Autoencoders

[17] Salakhutdinov, R., Hinton, G. E. (2009). Deep Boltzmann machines. In AISTATS'2009 – a paper introducing Deep Boltzmann machines

[18] Salakhutdinov, R., Mnih, A., and Hinton, G. E, Restricted Boltzmann machines for collaborative filtering, In ICML, 2007 – a paper introducing – a paper about modeling tabular data on the Netflix movie ratings dataset, using Restricted Boltzmann Machines

[19] Bossard, L., Guillaumin M., Van Gool, L., "Food-101 – Mining Discriminative Components with Random Forests", Computer Vision Lab ETH Zurich, 2014 – source of the Food-101 dataset

[20] Girshick, R., Donahue, J., Darrell, T., Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation,"Computer Vision and Pattern Recognition 2014 IEEE Confrence -

[21] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., "Imagenet large scale visual recognition challenge", published in ArXiv, URL: https://arxiv.org/pdf/1409.0575.pdf, 2014 – a paper that introduced the ImageNet dataset

[22] He K., Zhang X,. Ren, S., Sun, J. "Deep Residual Learning for Image Recognition", published in ArXiv, URL: https://arxiv.org/pdf/1512.03385.pdf , Dec 2015 – a paper that influenced the WISeR network

[23] He, K., Zhang, X., Ren, S., Sun, J. "Deep Residual Learning for Image Recognition", published in International Conference on Computer

Vision and Pattern Recognition, 2016 – a book that influenced the WISeR network

[24] Matsuda, Y., Hoashi, H., Yanai. K., "Recognition of multiplefood images by detecting candidate regions", IEEE International Conference on Multimedia and Expo (ICME), 2012 – paper on the recognition of multiple food items on a single image

[25] Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., "Deep learning adaptive computation and machine learning " (Vol. 1). Cambridge: MIT press, 2016 – a good introduciton to deep neural networks

[26] Duchi, J., Hazan, E., Singer, Y., "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", COLT, 2010 – a paper introcuding the Agagrad optimization method

[27] Kingma, P. D., Ba, L. J., "Adam: A Method for Stochastic Optimization", ICLR 2015 – a paper introducing the Adam optimization method

[28] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, 2014 – a paper introducing SIFT

[29] Roberts, C. M., "Radio frequency identification (RFID)", Computers & Security, Volume 25, Issue 1, February 2006, Pages 18-26 – a paper about RFID

[30] Cortes, C., Vapnik, V., "Support-vector networks", Machine Learning, Volume 20, Issue 3, pp 273-297, September 1995 – a paper about SVN

[31] Sharma, A., "Understanding Activation Functions in Neural Networks", The Theory of Everything, 2017 – a brief report about ReLU, Sigmoid and Tanh activation funciton