Contents

Spis treści

1 General information

This chapter is a summary of the most important parts of presented engineering thesis, which are subsequently explained in detail.

1. **Problem definition.**
   Acceleration of user access to beer database using computer vision and mobile device.
2. **Practical purpose.**
   Design and implementation of server-client recognition system for beer labels.
3. **Cognitive purposes.**
   (a) Studying impact of convolutional neural network (CNN) architecture and data augmentation on accuracy and training time of the classifier.
   (b) Gaining deeper understanding of computer vision using CNNs.
   (c) Finding minimum number of training iterations and images per class (per beer) to achieve at least 85% accuracy of the classifier.
4. **Scope.**
   (a) Developing a script for data augmentation.
   (b) Developing a script for performing bottleneck features extraction and training of convolutional neural network.
   (c) Developing RESTful server for providing label predictions.
   (d) Developing Android application for capturing and sending images to predict to server.
5. **Thesis.**
   Implemented convolutional neural network, classifying beer labels, is scalable and can achieve at least 85% accuracy under user-friendly conditions, with inference time not exceeding 1 second.

2 Problem definition

Report on the alcoholic beverages market in Poland, performed by audit company KPMG [21], claims, that in 2013, Poles spent 41.1 billion zlotys on alcohol, with beer being a large portion of that amount (47%). The report also states that, as of 2014, Polish customers choose beers based on premiumisation (31% of customers), innovation (29%) and region affiliations (30%), with manufacturers predicting significant rise of interest in those attributes in the future. The evolution in customers' taste and their openness resulted in many local shops offering a vast range of, so called, "craft beers" coming from lesser known breweries. However, it's also not uncommon to stumble upon large product shelves, with dozens different beers types, in more popular Polish stores.

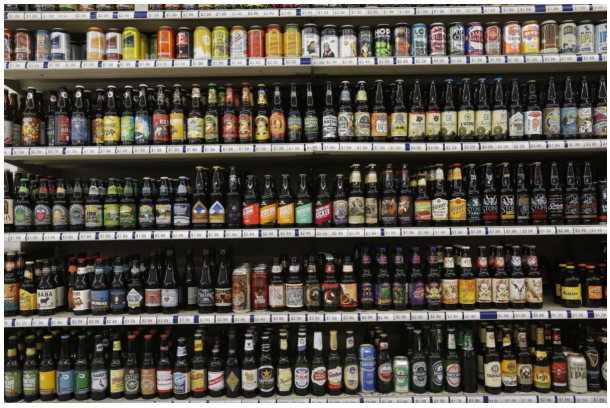

Figure 1: A large beer shelf. Source: `http://trib.com`.

The question this thesis posed at first was "How technology could be used to help a customer, who stands in front of a supermarket's large beer shelf (Figure 1.) and tries to choose one, not knowing anything about beers?". The answer appeared to be relatively simple and involved developing a mobile application, which could connect to beer database residing in Cloud e. g. Firebase Realtime Database and retrieve details about any given beer. The real problem, that later emerged, and is in fact the basis of this thesis, is "Given a mobile application, that can connect to beer database in Cloud, what is the most intuitive, appealing and fastest way of accessing this database from the device?".

3 Solution overview

To further investigate the problem articulated in Chapter 2, a survey was conducted [15]. Its main question was "Imagine standing in front of a big shelf with hundreds of craft beers you've never seen before. There's no one there to help you except a mobile app with beer database, which you can access in different ways: typing beer name, using voice to input beer name, making photo of front beer label, making photo of bar code in the back of the bottle. Rank those ways starting from the one that's the most intuitive and appealing to you.". Interestingly, the most popular option, among 50 people asked, was making a photo of front beer label. This method of database access was almost equally popular to the most standard human-device communication interface, which is keyboard (typing a beer name). Whereas Figure 2 presents the overall trend, Figure 3 provides us with detailed numbers: 60% of people chose, in either first or second place, making a photo of front beer label as their preference (same as in typing a beer name by hand). The last thing that's worth noting is the lack of popularity of barcode scanning.

Based on the aforementioned survey and personal experience, a solution to the problem, stated in Chapter 2, was formulated as a design and implementation of server-client recognition system for beer labels.

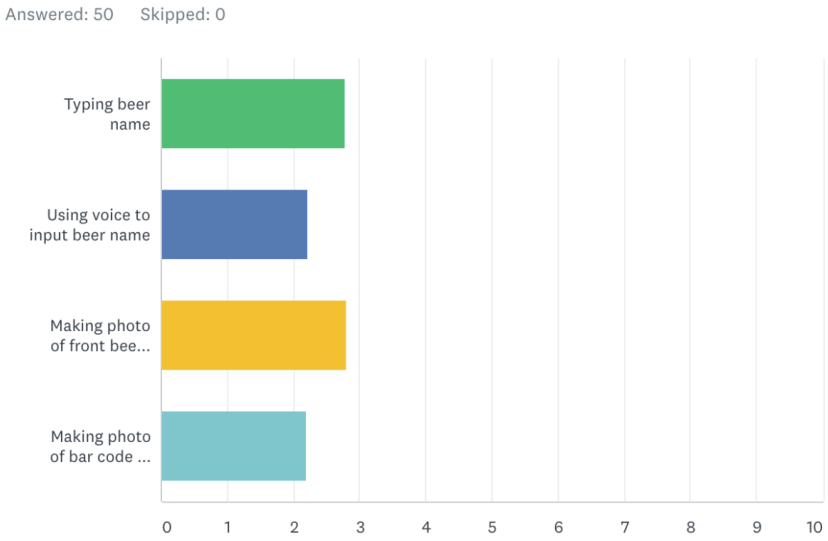

Figure 2: Weighted average of respondents' answers (chart). For choosing an option as first - weight: 4, for choosing an option as second - weight: 3 etc. Obtained sum is then divided by total number of respondents (here: 50).

The goal of below section is to give a reader, in a few short points, an overall idea of the recognition system before theoretical introduction. Chapter 11 focuses on implementation details.

Figure 4 presents a scheme with a functional principle of implemented server-client beer label recognizer, namely:

1. An image of beer label is taken by the Android [12] application.
2. Android application sends the image to the Flask [14] server equipped with algorithm, which performs beer label prediction.
3. Encoded prediction is returned to Android application.
4. Encoded prediction is used as a key to the beer database residing in Firebase Cloud [13].

5. Based on the key (encoded label name) all information about given beer is sent to Android application and displayed to the user.

	1	2	3	4	TOTAL	SCORE
Typing beer name	32.00% 16	28.00% 14	26.00% 13	14.00% 7	50	2.78
Using voice to input beer name	28.00% 14	10.00% 5	18.00% 9	44.00% 22	50	2.22
Making photo of front beer label	30.00% 15	30.00% 15	30.00% 15	10.00% 5	50	2.80
Making photo of bar code in the back of the bottle	10.00% 5	32.00% 16	26.00% 13	32.00% 16	50	2.20

Figure 3: Weighted average of respondents' answers (table with details). For method of average calculation see: Figure 2.

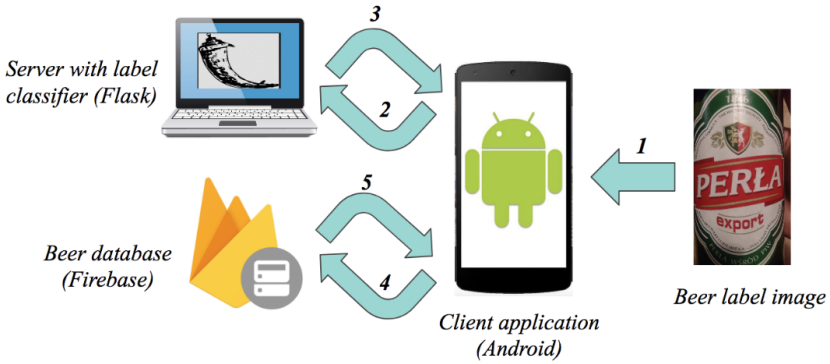

Figure 4: Server-client recognition system for beer labels with functional principle (consecutive steps enumerated).

4 Related solutions

To assess a value of the proposed solution to acceleration of access to beer database problem, a research into available related techniques was conducted. It is worth noting that there seems to be no popular and continuously maintained application solving the issue using deep learning.

1) SIFT algorithm.
   Scale-Invariant Feature Transform algorithm was first described by Lowe in 1999 [23] and was used for beer label recognition by Weitz et al. at Stanford University [27]. The technique extracts local features from image and later uses them for classification. SIFT proved to be robust to uniform scaling, orientation and illumination changes. It is, however, protected by patent (additional costs). Lowe's algorithm is considered mathematically complicated and computationally heavy. Each beer label identification using this solution took about 10 seconds on 8 parallel processors in MATLAB (again, more computational power imposes more costs). SIFT is included in popular OpenCV library.

2) Bar code and QR code scanners.
   Ohbuchi et al. [25] presented methods capable of very effective code scanning using mobile devices, which are now used across many fields for fast data retrieval. The amount of information stored in such codes is limited and defined by global norms (e. g. 1D bar code obeying Code 39 standard is advised not to contain more than 25 characters, while QR code limits are much higher - up to 1520 alpha numeric characters). Untappd is a social beer drinking mobile app, which uses bar code scanning to access its database. Despite this technique being very reliable and accurate, it creates a challenge of staying up to date with all bar code changes implemented by brewery, that oftentimes take place during some promotional campaigns. Bar code scanning also requires user to physically search for bar codes on a bottle and lacks the "fun" factor of image-based searches. QR codes are still not widely used for product identification.

3) OCR.
   Vivino is a mobile application with access to world's largest wine database. Its algorithm used for obtaining data is based on Optical Character Recognition, namely, keywords are being extracted from wine label photo and used as an input to app's search engine. The highest costs in Vivino's approach are associated with a need to incorporate third party OCR provider (e. g. ABBYY) and creation of database supporting advanced partial searching.

4) Custom image recognition API provider.
   Vize.ai is an example of service allowing to easily create custom recognizer of specific images. It abstracts all complexity related to designing own classifier, training and uploading it to remote server. On the contrary, one of the cognitive purposes of this thesis was to delve into image recognition complexities and to obey MVP rules (Minimum Viable Product) articulated, among others, by Ries [26], hence third party recognition provider was not taken under consideration.

5 Introduction to artificial intelligence

There are many approaches to define Artificial Intelligence (AI), but perhaps the most famous is the one formed by Alan Turing in 1950. The proposed method, referred to as Turing test, involved human subject asking written questions to a computer and receiving written answers. If the interrogator couldn't tell if responses came from a human or a machine - the test was passed, i. e. the computer was considered intelligent (at least in some sense). Even stricter rules were imposed on conduction of, so called, total Turing test. In addition to requirements mentioned previously, a computer needed to fool a human subject through a proper visual perception and human-like object manipulations (robotics).

Achieving a feat of passing either of aforementioned tests requires multidisciplinary knowledge in many fields. Norvig and Russell [24] did a remarkable job of listing major disciplines, that contributed to the growth of AI, namely:

a) philosophy,

b) mathematics,

c) economics,

d) neuroscience,

e) psychology,

f) computer engineering,

g) control theory and cybernetics,

h) linguistics.

Despite invaluable contributions from greatest minds in above fields, the Holy Grail of some researchers - Artificial General Intelligence, often called "strong AI", capable of performing any intellectual task of human being - is still far ahead. Focus of many scientists, and also of this thesis, is, conversely, on creating algorithm, which behaves in an intelligent manner in a specific area of expertise (e. g. computer vision), but fails miserably in others (e. g. voice recognition). A big part of this approach revolves around a branch of Artificial Intelligence called supervised learning.

In essence, supervised learning is inference of correct outputs from given inputs based on labeled training set. Examples of this include:

a) given an email, is it spam or not?

b) given an image, does it contain face or not?

c) given an audio clip, what is the correct transcript?

d) given a Polish sentence, what is semantically correct English translation?

e) given a beer label image, what is the name of the beer?

Prerequisites of performing such tasks are, usually, a large amount of data (e. g. actual emails correctly marked as spam or not) and big computational power (powerful Graphics Processing Units, High-Performance Computing), both of which significantly increased over the past years (Figure 5, Figure 6).

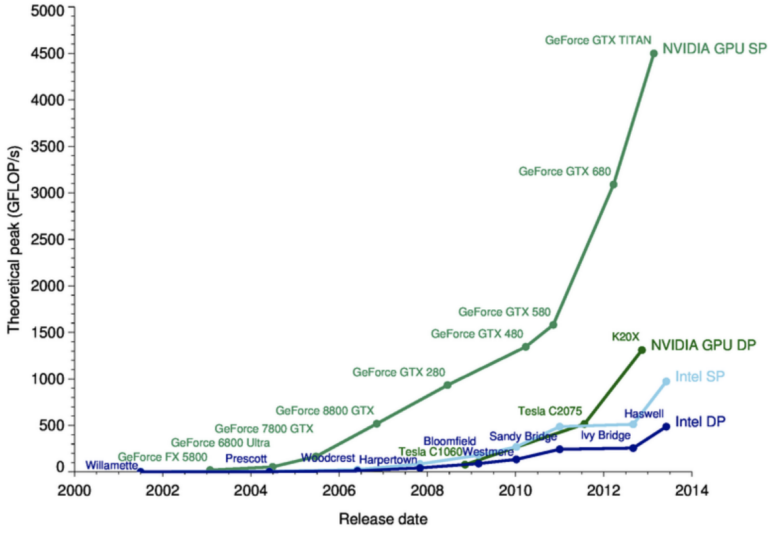

Figure 5: Performance of NVIDIA GPUs and Intel CPUs measured in GFLOP/s [16].

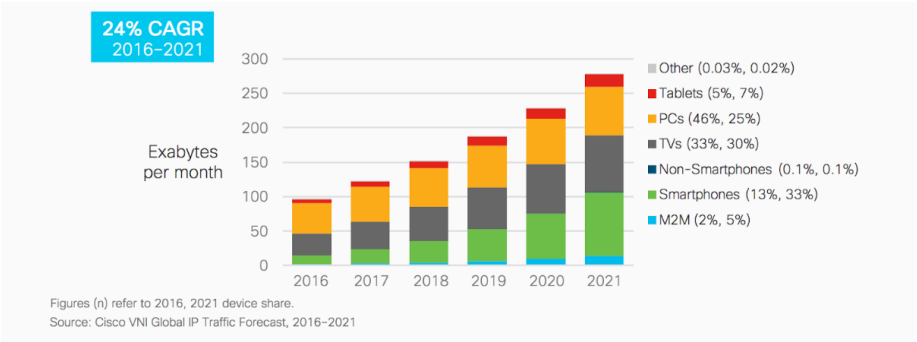

Figure 6: Global IP traffic by devices [20] with 24% compound annual growth rate (2016-2021).

It is important to distinguish between Machine Learning (ML) and Artificial Intelligence. The former is considered to be a subset of the latter. Whereas AI comprises of knowledge base systems, genetic algorithms, Bayesian statistics etc., none of that is considered ML. Machine Learning can be defined as an act of optimization along a certain dimension (e. g. minimizing the error of choosing a wrong prediction of a beer label based on an image) and a primary tool for that is an algorithm called neural network, loosely based on a human brain cell.

6 Introduction to neural networks

It's reasonable to think of a neural network (NN) as a mathematical function, which in practice tends to be very complicated because of three things:

1) it has a large number of coefficients (weights), often exceeding tens of millions,

2) it's a very deeply nested function, hence even simple gradient calculation (partial derivative) is relatively slower,

3) most of its computations are performed on multidimensional tensors.

Figure 7 contains a popular representation of a simple neural network with three basic building blocks: unit (single circle with value $in_i^{(k)}$, input x_i, output $\hat{y_i}$ or bias 1), layer (units arranged into one vertical group k) and weight (connection between units with value $w_{ij}^{(k)}$ representing its strength). Equation 1, 2, 3, 4 and 5 translate this graphical representation to mathematical formula.

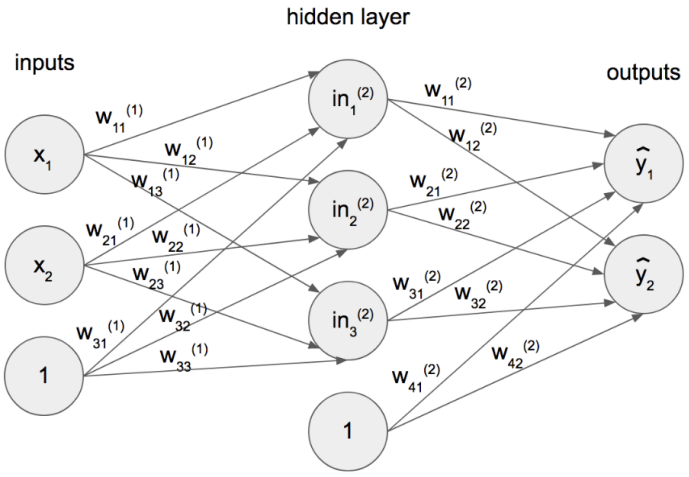

Figure 7: A simple neural network with one hidden layer (a layer between inputs and outputs).

$$\hat{y_1} = w_{11}^{(2)} in_1^{(2)} + w_{21}^{(2)} in_2^{(2)} + w_{31}^{(2)} in_3^{(2)} + w_{41}^{(2)} * 1 \tag{1}$$

$$\hat{y_2} = w_{12}^{(2)} in_1^{(2)} + w_{22}^{(2)} in_2^{(2)} + w_{32}^{(2)} in_3^{(2)} + w_{42}^{(2)} * 1 \tag{2}$$

$$in_1^{(2)} = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} * 1 \tag{3}$$

$$in_2^{(2)} = w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{32}^{(1)} * 1 \tag{4}$$

$$in_3^{(2)} = w_{13}^{(1)} x_1 + w_{23}^{(1)} x_2 + w_{33}^{(1)} * 1 \tag{5}$$

"Perceptron" is a common name for a neural network, where inputs are immediately coupled with outputs (no hidden layers, unlike in Figure 7). The presence of hidden (middle) layer of units, which prevents from direct connections between inputs and outputs, allows neural network to model highly nonlinear mathematical functions. Norvig and Russell justify that, using XOR gate as an example, in a following manner: "[...] linear classifiers [...] can represent linear

decision boundaries in the input space. This works fine for the carry function, which is a logical AND [...]. The sum function, however, is an XOR (exclusive OR) of the two inputs. [...] this function is not linearly separable so the perceptron cannot learn it. The linearly separable functions constitute just a small fraction of all Boolean functions." [24]

Before delving into learning process of NNs, it's important to make two additions to previous model:

1) error function (also called cost function),

2) activation function.

Ad 1. The most reliable way for the algorithm to represent predictions is through a vector of probabilities. Consider an example of beer name predictions based on image of label. Figure 8 shows a probability output of a classifier (notice that all values sum to 1), compared with an output, that it should strive for. A cost function, introduced in this section, called categorical cross entropy (Equation 6), simply measures the correlation between those two probability distributions (predicted and ideal). Notice that multiplication by one-hot encoded examples, forces the function to only compare non-zero elements of ideal distribution, with respective values of classifier output further from 1 being penalized more than values close to 1 (thanks to the nature of logarithm).

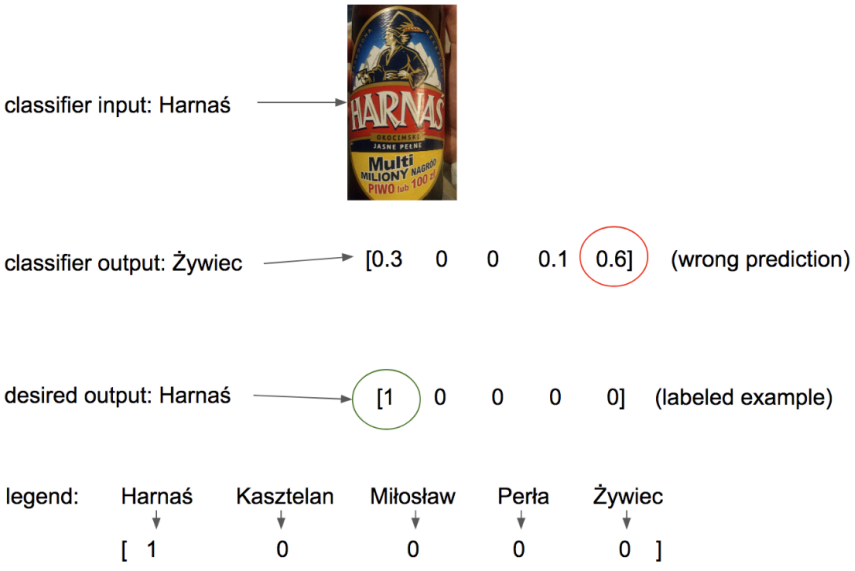

Figure 8: Exemplary classifier input, output and desired (one-hot encoded) output with legend.

$$H(d,p) = -\sum_i d_i * log(p_i) \tag{6}$$

where:

d_i is the i^{th} element of one-hot encoded (desired) probability vector d,

p_i is the i^{th} element of probability vector p predicted by the classifier.

Ad 2. Unit's value $in_i^{(k)}$ is rarely propagated explicitly to next layers. So called activation function is used instead. The one introduced in this section is called sigmoid (Equation 7). The

updated model of simple neural network from Figure 7 is shown in Figure 9. One thing worth pointing out is a difference between sigmoid and softmax function (Equation 8) - both used in artificial neural networks. Whereas sigmoid inputs a single value and outputs a normalized scalar, softmax inputs a list of values and outputs a vector of real numbers in range [0, 1] that add up to 1, thus can be interpreted as a probability distribution. Sigmoid is used in hidden units, while softmax is usually applied in the last output layer. Both functions can be categorized as logistic functions.

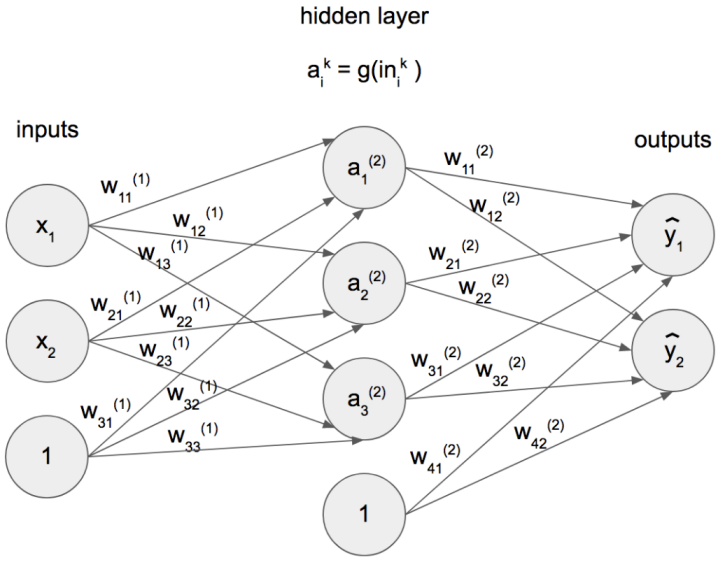

Figure 9: Updated model of simple neural network from Figure 7. g is a sigmoid activation function. Outputs are often normalized using softmax.

$$g(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

$$s(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{8}$$

Softmax function $s(z)_j$ "squashes" a K-dimensional vector z to K-dimensional probability distribution, where $j = 1, ..., K$.

The goal of neural network's learning process is to find correct weights, i. e. weights, that will result in a mathematical model, where the difference of inputs is clearly represented in the difference of output vectors, which are subjects to analysis and prediction. For example in a trained dog breed classifier, the output vector for an image of german shepherd is clearly different than for york's. This can be easily interpreted and lead to correct human-readable prediction of a breed. Currently, the best known way to train a network is via algorithm called backpropagation. Main idea of this method is to calculate gradients of a cost function E (e. g. categorical cross entropy) with respect to each of weights, which are later updated by some portion of these gradients as illustrated in Equation 9.

$$w_{ij}^{'(k)} \leftarrow w_{ij}^{(k)} - \alpha \frac{\partial E}{\partial w_{ij}^{(k)}} \tag{9}$$

where:

α is a learning rate (indicating what portion of gradient should be used).

Let us consider a neural network in Figure 10 with three units, one hidden layer and sigmoid activation function. Before conducting backpropagation, so called, forward pass was performed, which simply is a mathematical inference of outputs, given inputs (Equation 10).

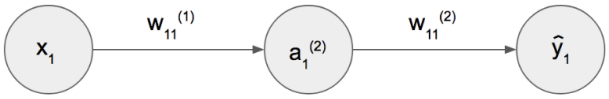

Figure 10: A simple neural network, with sigmoid activation function g (Equation 7), i. e. $a_1^{(2)} = g(w_{11}^{(1)} x_1)$.

$$\hat{y_1} = \frac{w_{11}^{(2)}}{1 + e^{-w_{11}^{(1)} x_1}} \tag{10}$$

As mentioned previously, NN's learning algorithm is based on calculating partial derivatives with respect to each of weights. A deep nesting of functions, representing more complicated networks, encourages to make use of chain rule [17]. Figure 11 outlines a single step of backpropagation using categorical cross entropy error function E. Equation 11 and Equation 12 present symbolic gradient calculations, necessary for learning process to occur. At this point, a beautifully simple derivative of sigmoid function is worth recalling:

$$s'(x) = \frac{\partial}{\partial x} \frac{1}{1 + e^{-x}}$$

$$s'(x) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

$$s'(x) = \frac{1}{1 + e^{-x}} * \frac{e^{-x}}{1 + e^{-x}}$$

$$s'(x) = s(x)(1 - s(x))$$

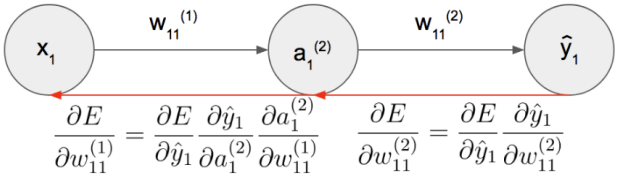

Figure 11: Outline of a single step of backpropagation with chain rule, where $E = -y_1 log(\hat{y_1})$. Value y_1 is a desired binary output from training set.

$$\frac{\partial E}{\partial w_{11}^{(2)}} = -\frac{y_1}{ln(10) * \hat{y_1}(1 + e^{-w_{11}^{(1)} x_1})} \tag{11}$$

$$\frac{\partial E}{\partial w_{11}^{(1)}} = -\frac{w_{11}^{(2)} y_1 x_1 e^{-w_{11}^{(1)} x_1}}{ln(10) * \hat{y_1}[(1 + e^{-w_{11}^{(1)} x_1})^2]} \tag{12}$$

With symbolic computations behind, consider following inputs to neural network from Figure 11:

1) $x_1 = 10, y_1 = 1$ - a training example. NN should strive for returning 1, via variable y_1, whenever the input is equal to 10.

2) $w_{11}^{(1)} = 0.3, w_{11}^{(2)} = 0.5$ - randomly initialized weights for first forward pass (Equation 10).

Primary step of learning process is performing inference, given randomly initialized weights and input. The produced outcome is $\hat{y_1} \approx 0.48$, which is quite far from desired 1 (y_1). Backward pass allows to calculate gradients with respect to each weight, namely $\frac{\partial E}{\partial w_{11}^{(2)}} \approx -0.86, \frac{\partial E}{\partial w_{11}^{(1)}} \approx -0.2$. After applying update rule from Equation 9, with learning rate $\alpha = 0.5$, new weights are $w_{11}^{(1)} \approx 0.4, w_{11}^{(2)} \approx 0.93$ and produce outcome $\hat{y_1} \approx 0.91$(much closer to desired 1). Presented algorithm is iterative. With increased amount of repetitions of above step and larger amount of examples, it should converge to optimal weights (globally or locally).

7 Introduction to convolutional neural networks

In Figure 12 a reader can see a common representation of a grayscale image, depicting number 7 in low resolution (4x4). Supplying these pixel values as a vector input of size 16 to a regular neural net, detecting if number 7 is present in a picture or not (Figure 13), could yield a desired result, but with a huge cost. All inputs must be mapped to each hidden neuron, hence the amount of weights will grow drastically for even slightly higher image resolution. Additional pitfall is a complete loss of spatial connections between pixels, which are essential during object recognition. The solution involves introduction of local connectivity. Consider breaking a picture from Figure 12 into four regions as in Figure 14. Then, each hidden node could be coupled with only the pixels from one of these regions as Figure 15 and Figure 16 show. This simple idea is utilized in convolutional neural networks (CNNs/ConvNets) and will assist during further comprehension of this chapter.

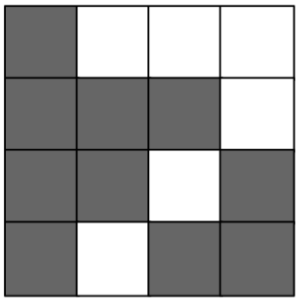

Figure 12: Grayscale image of number 7 (4 x 4 pixels resolution).

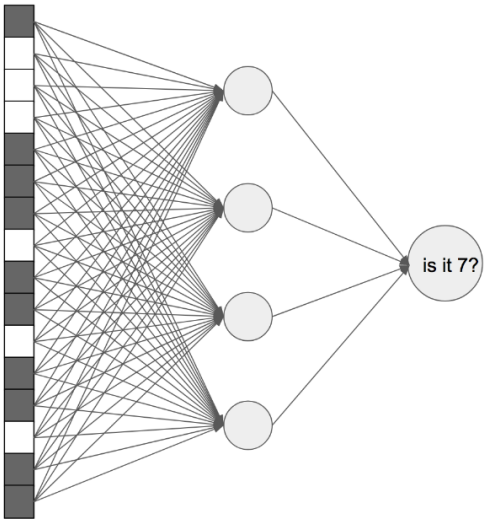

Figure 13: Regular neural network recognizing number 7 in image input from Figure 12, which is deconstructed (flattened) into vector of size 16.

The standard reference for ConvNets is *"Object Recognition with Gradient-Based Learning"* by LeCun et al. (1999) [22]. This section attempts to explain 2 most basic building blocks of the system designed in that paper, called LeNet-5 (Figure 17), which are:

1) convolutional layer,

2) subsampling layer.

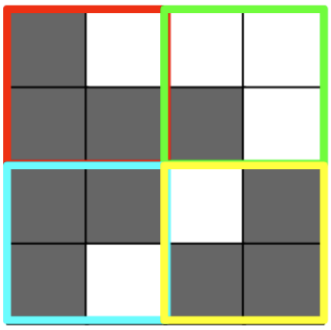

Figure 14: A number from Figure 12 divided into four pixel regions.

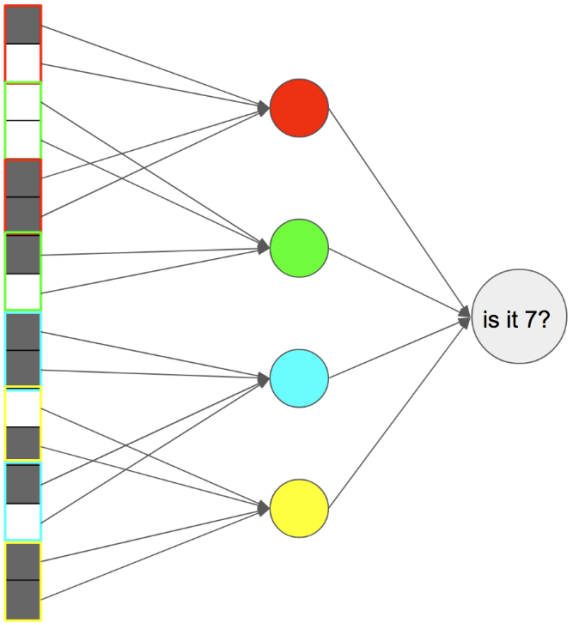

Figure 15: Local connectivity of inputs from Figure 14 to hidden nodes (vector form).

Ad 1. Filter (also called convolution window) is a tool for obtaining convolutional layer. In essence, it's a matrix with coefficients (weights), which slides, with a certain stride, over previous layer (in Figure 18 the previous layer is an input layer) and performs a dot product. The outcome of this operation is saved into a new matrix, called a feature map. A collection of features maps (in case of Figure 18 a single feature map) is called a convolutional layer. There are only a few major differences between simplified model above and the ones used in practice. Firstly, pixel values are non-binary, i. e. they are represented by an integer between 0 and 255, which is often normalized to fall into a real range [0, 1]. Secondly, recognition is often performed on RGB images (Red Green Blue [18]). As a result of that, the size of the input is width x height x depth instead of width x height, where depth simply equals the number of color channels: 3. Convolutions are then performed simultaneously across each color matrix. Thirdly, Figure 18 presents only a single, atomic step constituting inference in CNNs. In next phase, the output of previous convolution (a collection of feature maps) is undergone the same process, but with a use of a different filter. And lastly, filters are often stacked together (multiple convolution windows are used at the same time). Only then, succeeding convolutional layer can possess multiple feature maps. As with regular neural networks, the goal of the learning process, using backpropagation, is to find correct weights (filter coefficients).

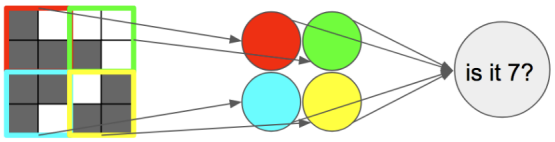

Figure 16: Local connectivity of inputs from Figure 14 to hidden nodes (matrix form).

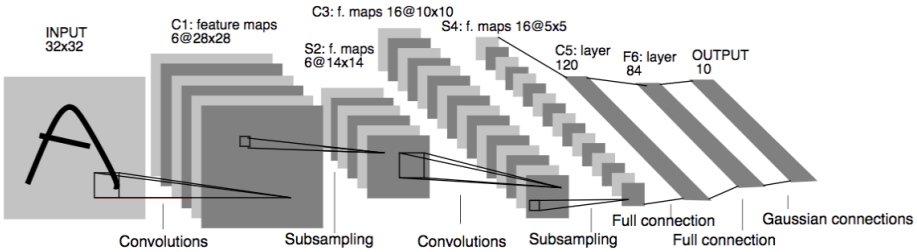

Figure 17: LeNet-5 CNN [22] used for handwritten digit recognition.

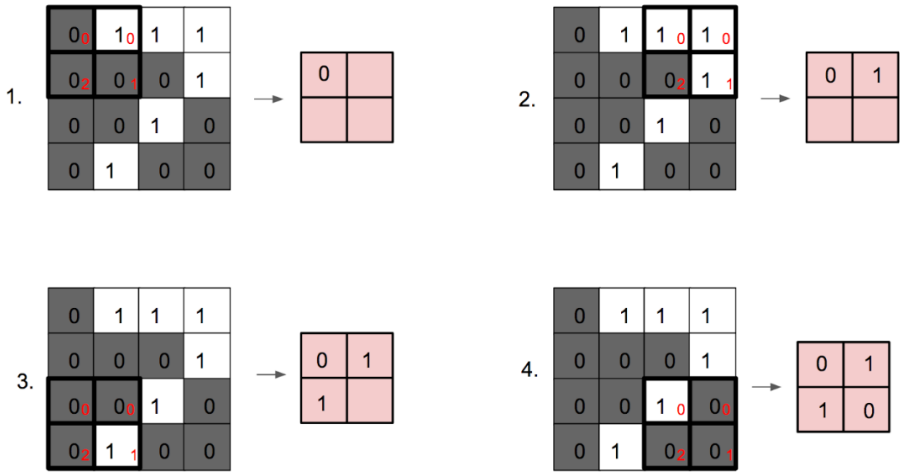

Figure 18: Consecutive steps of convolution using filter of size 2x2 with stride 2 (coefficients visible in red in the bottom right corner of a filter). The pink matrix is called a feature map and in this case represents a convolutional layer.

Ad 2. Subsampling layers serve two purposes: reducing network dimensionality and helping it to generalize. One of the most popular subsampling techniques used today is max pooling. A window slides across a matrix with a certain stride (in a way similar to filter), takes element with the highest value and saves it to a reduced tensor (Figure 19).

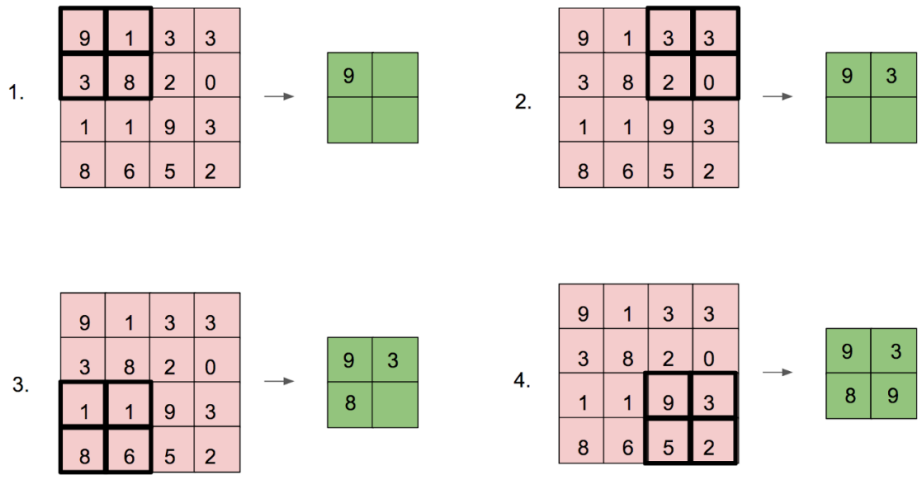

Figure 19: Consecutive steps of max pooling with window of size 2x2 and stride 2. Green matrix is a reduced tensor.

Last thing worth mentioning is that, similarly to normal neural networks, CNNs rarely propagate activations (values of elements in features maps) explicitly to next layers. Rectifier function, also called ReLU (Equation 13), is often used instead.

$$f(x) = max(0, x) \tag{13}$$

8 Comparison of world-class CNN architectures

Convolutional neural networks, described in the previous chapter, take significant amount of training data to achieve high level accuracy. It is not uncommon, for more complicated recognizing tasks, to require hundred thousands of labeled images. One of the websites, which hosts such data, is ImageNet. As of November 4, 2017, it contains 14197122 images with 21841 synsets (i. e. groups of synonyms helping with information retrieval [11]). However, the greatest merit of ImageNet is hosting ILSVRC competition, that gave rise to CNNs, which are now considered world-class. One of the tasks of the competition is to implement an algorithm capable of recognizing 1000 different objects and drawing bounding boxes around them, given 1.2 million training images. Canziani et al. [19] noticed, that because of the competition's focus solely on accuracy, other metrics, important for practical applications, are neglected. In Figure 20 a reader can see a comparison of the most popular ILSVRC entries in the last years, which provides information about:

1) Top-1 accuracy - obtained by checking if the top class (the one with the highest probability in the output vector of ConvNet) is the same as target label.

2) Operations - number of operations required for inference of a single image. Linearly related to the time of a single image classification process.

3) Parameters - illustrated by circle size. The amount of neural network's parameters i. e. the size of NN or a measure of how complicated a model is.

Furthermore, authors of the comparison used standardization not only by task and training set, but also by hardware used: "All experiments were conducted on a JetPack-2.3 NVIDIA Jetson TX1 board (nVIDIA): an embedded visual computing system with a 64-bit ARM A57 CPU, a 1 T-Flop/s 256-core NVIDIA Maxwell GPU and 4 GB LPDDR4 of shared RAM." [19].

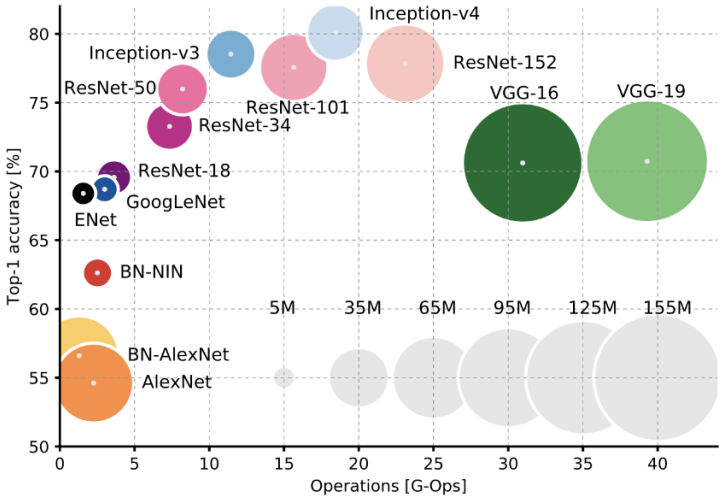

Figure 20: Comparison of world-class CNN architectures [19] including: top-1 accuracy, number of operations and parameters.

9 Transfer learning

The state-of-the-art CNN architectures, mentioned previously, are result of extensive experimentation, hyperparameters tuning, many years of expertise and months of dedicated hard work. They were based on ImageNet database and often took weeks to train. How then, could one adapt those advanced architectures to own classification tasks? The answer lies in technique called transfer learning (TL). In essence, TL is used for merging two neural networks: one, that is quite accurate at some task (e. q. everyday object recognition) and the other, which is mediocre (e. g. 5% accuracy at beer label recognition). Thanks to the nature of the connection, a worse NN uses expertise of the better one, thus increasing own accuracy. This chapter focuses on the most common type of transfer learning, that is given a small data set, which is similar to the one advanced CNN was trained on.

Transfer learning step by step:

1) Slice off the last fully connected layer of advanced neural network, that you base TL on (Figure 21).

2) Perform inference of your entire training set, using advanced NN with sliced layer and pre-trained weights, ending up with, so called, bottleneck features i. e. a tensor image representations created by a given CNN (Figure 22).

3) Perform training on your second (mediocre) neural network using labeled bottleneck features as inputs (Figure 23).

Above technique could differ depending on below cases:

a) small data set, which is different than the one advanced CNN was trained on,

b) large data set, which is similar to the one advanced CNN was trained on,

c) large data set, which is different than the one advanced CNN was trained on.

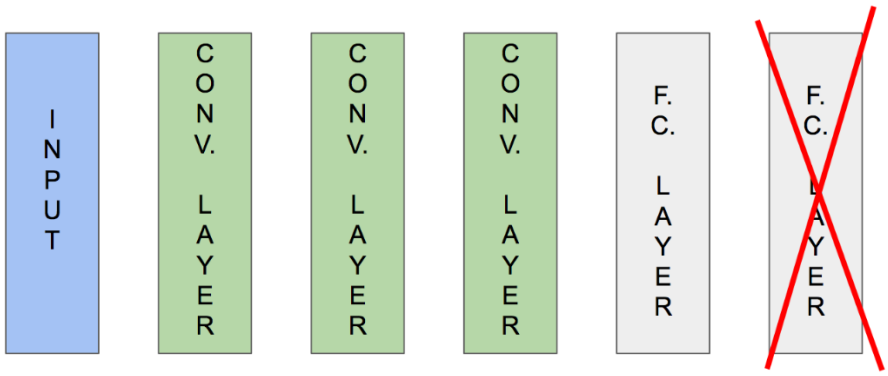

Figure 21: Advanced CNN without last fully connected layer.

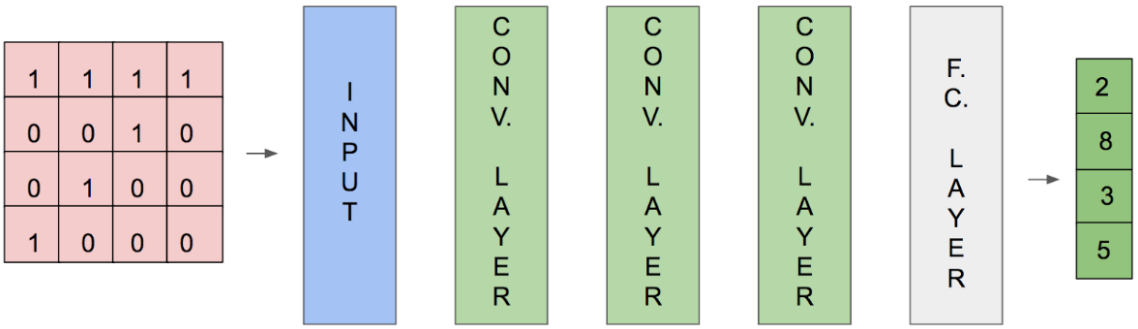

Figure 22: Inference of one image (pink matrix) from training set using advanced CNN with sliced layer and pre-trained weights. The output (green vector) is a bottleneck feature i. e. this CNN's representation of a given image.

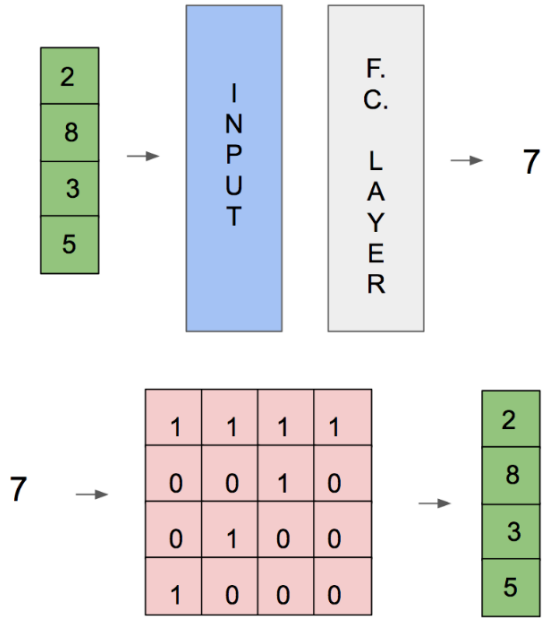

Figure 23: Inference in second (mediocre) neural network used in transfer learning, which is trained on labeled bottleneck features, with labeling illustrated below.

10 Data augmentation

One can deduce, from previous introduction chapters, that an important requirement for achieving accurate machine learning models is a lot of data. Whereas for more general recognition tasks World Wide Web is a place to search for training images, the more specific the task the higher possibility of manual data collection. In that case, data augmentation is a helpful technique, allowing to artificially inflate small data set by applying a series of transformations and saving result as a new picture. Figure 24 presents exemplary augmentation of a single picture of STOP sign, yielding 10 additional training data points.

Figure 24: Data augmentation applied on a single STOP sign image.

Transformations performed on images using data augmentation include:

a) changing brightness - involves converting images from RGB color space to HSV [5] and adjusting V value,

b) adding noise - imposition of Gaussian noise,

c) blurring - again, Gaussian blur is oftentimes used,

d) rotation - within specific angle range,

e) translation - within specific distance range,

f) shearing - displacement of each image point, according to Equation 14 and 15, resulting in distortion.

$$(x', y') = (x + my, y) \tag{14}$$

Formula for horizontal shear of points in 2D space.

$$(x', y') = (x, mx + y) \tag{15}$$

Formula for vertical shear of points in 2D space.

11 Solution walkthrough

Focus of this chapter is reexamination of Figure 4, with emphasis on used technology and implementation details.

1) Client application (Android) [1].
   This part is a basic interface used for user-server communication. The app has ability to take pictures, using built-in system camera, and then send them to server via software library called Retrofit [9] with a popular RxJava adapter [10]. The application is written in Kotlin programming language, which, as of mid 2017 [6], is officially supported on Android by Google. Kotlin turned out to be much more effective, expressive and less verbose alternative to Java, with which it is interoperable.

   Android app was developed from scratch in Android Studio integrated development environment (IDE) utilizing object-oriented and functional programming paradigms. Application's main component is Activity, which represents a screen that user can interact with (Figure 25). Clicking on camera button at the bottom of main screen results in redirection to device's camera. After successful capture of label photo, proper information about classified beer is shown.

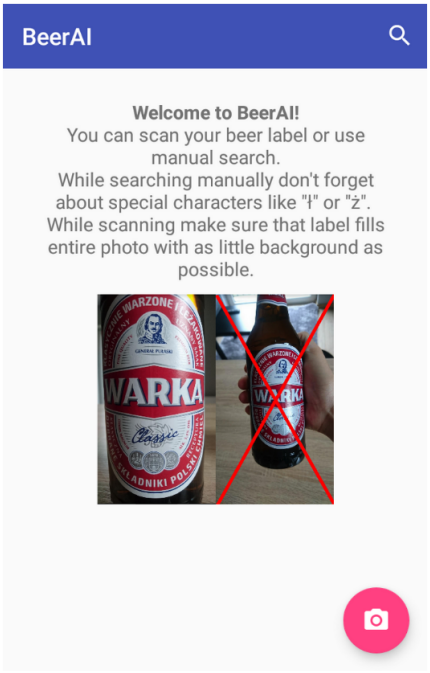

Figure 25: Main screen of Android application. Note the hints given to the user, resulting from prototype's limitations, namely: poor support of advanced searching in Firebase Realtime Database and high risk of wrong prediction given beer label image with a lot of background.

2) Beer database (Firebase).
   This part is a database with beer information, which is sent to Android application after successful recognition. It resides on Google's NoSql cloud server [7]. Data is stored as JSON and can be retrieved using very intuitive key-value query system. Despite being simple to launch and maintain, Firebase Realtime Database has currently very poor support for partial query searching, which is very anticipated in the upcoming releases.

Data about each beer were entered manually using Firebase console. Each beer is represented as a parent node (Figure 26) and contains two child nodes with keys: "description" and "name", which are encapsulated into object and sent to Android client after successful recognition.

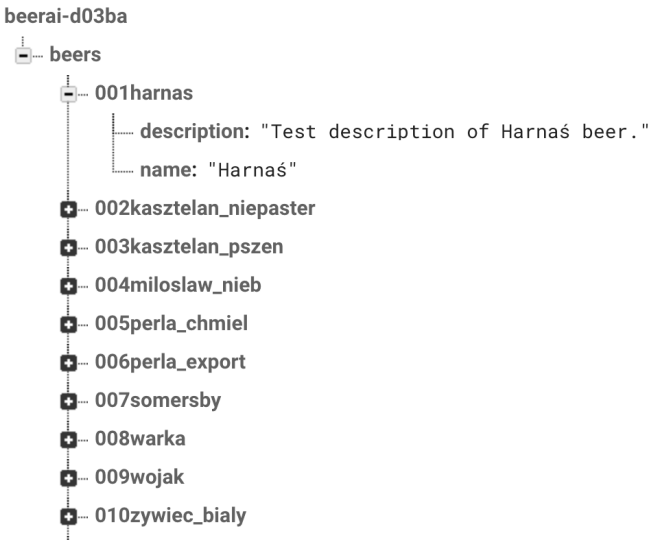

Figure 26: Firebase Realtime Database with encoded key-value pairs of beer information.

3) Server with label classifier (Flask) [2].
This part is a web server, with trained convolutional neural network, acting as beer label classifier. It is implemented from scratch in Flask - a Python microframework used for launching REST [8] servers. The endpoint responsible for prediction is called "/predict" and requires an image file from the caller (e. g. Android application). "/predict" obeys REST standard by using two HTTP methods: GET (as it returns encoded prediction of beer) and POST (as it accepts image file from client as input).

```
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.1.114 - - [18/Nov/2017 15:58:22] "POST /predict HTTP/1.1" 200 -
192.168.1.114 - - [18/Nov/2017 15:58:29] "POST /predict HTTP/1.1" 200 -
192.168.1.114 - - [18/Nov/2017 15:58:38] "POST /predict HTTP/1.1" 200 -
```

Figure 27: Terminal view of requests sent to Flask server (to "/predict" endpoint) from Android application. "200" return code indicates that request succeeded.

4) Trained convolutional neural network from Flask server [4] [3].
This part is a CNN created exclusively for beer label recognition on Flask server. To create training data set, 2 - 3 pictures of labels were collected per beer. Then data augmentation technique was used, resulting in 300 - 450 images per class. Applying transfer learning created connection between two neural networks - the "worse" one, which is presented in Figure 28 and the "better" one - Inception-V3 - chosen from Figure 20, due to its small size, quick inference and relatively high accuracy. Entire procedure was implemented using Keras library with TensorFlow backend (in Python programming language). Next chapter provides reader with detailed metrics about obtained ConvNet.

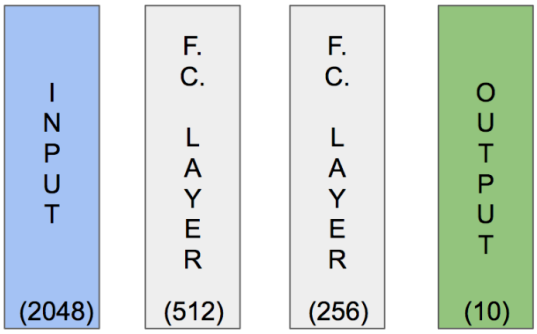

Figure 28: Regular fully-connected neural network, which accuracy increased due to transfer learning based on Inception-V3. Values in brackets represent amount of neurons in each layer. Whereas number of neurons in hidden layers (f. c. layers) is fixed, inputs and outputs depend, respectively, on size of bottleneck features coming from Inception-V3 and number of beers CNN can recognize (in basic version of prototype it's 10). Both hidden layers use ReLU activation function.

12 Testing

Purpose of this chapter is to provide reader with metrics related to the convolutional neural network used in the prototype, which is the base of this thesis. Those metrics will be later utilized for system evaluation. All experiments were conducted on 3.1 GHz Intel Core i5 CPU with 16 GB 2133 MHz LPDDR3 RAM.

Figure 29 presents manually collected data set for one class. Augmentation, applied on this beer label, resulted in 150 additional pictures per image, i. e. 302 images total for Harnaś beer, and took on average 256.91 seconds to complete. Similar procedure was employed for remaining 9 beer labels. Figure 30 contains a small portion of augmented training set. Picture transformations used:

a) changing brightness - for every piece of data (using HSV color model),

b) adding Gaussian noise - 20% probability of happening,

c) Gaussian blurring - 20% probability of happening,

d) rotation - for every piece of data (max angle value: 90 deg),

e) translation - for every piece of data (in both directions),

f) shearing - 25% probability of happening.

Figure 29: Manually collected data set for Harnaś beer label class. The first picture (2160 x 3840 pixels resolution) was taken by camera and the second one (ideal) was found on the internet.

Figure 30: Small portion of augmented training set for Harnaś beer label class.

Bottleneck features were extracted, using Inception-V3 neural network, for 3020 training images of all beer labels, that the prototype is capable of recognizing. At the same time, each picture underwent a dimension reduction to 150 x 150 pixels resolution and value rescaling to real range

[0, 1]. This procedure took 428.19 seconds for entire data set.

Categorical cross entropy error function was used during training. Number of epochs (learning iterations) was set to 50, while weights were being updated after backward pass on batches of size 48. To validate accuracy a validation set of beer labels images was created, independently from the training one, and fed to obtained ConvNet (over 100 validation pictures). Figure 31 and 32 contain charts of training progression, which in the end took 767.3 seconds for 10 classes and resulted in learning a total of 9571594 parameters.

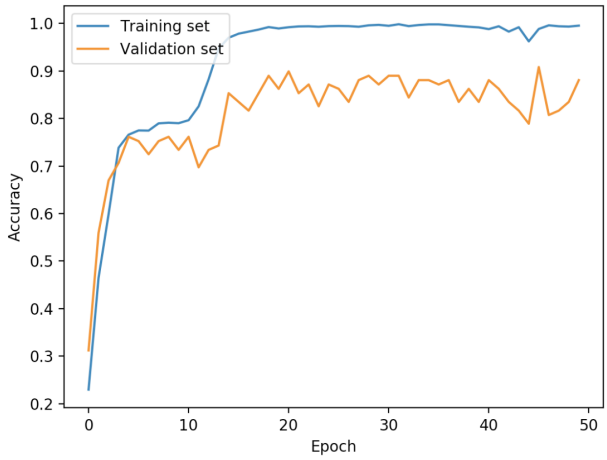

Figure 31: Accuracy of 10-class model during training. The best obtained value is 99.5% for training set and 89.1% for validation set.

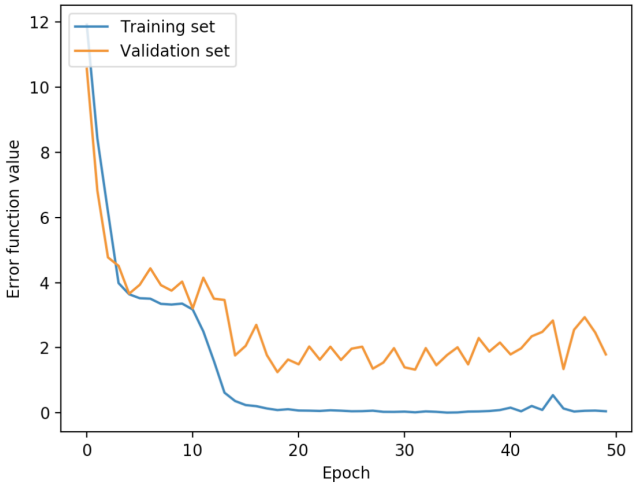

Figure 32: Categorical cross entropy error during training of 10-class CNN. The best obtained value is 0.051 for training set and 1.796 for validation set.

Average time of a single beer label recognition, out of 10 classes, is 0.1 second. Implemented CNN was tested in different lighting conditions and with or without visible background in input image. Figure 33 accumulates outcomes of these tests.

Taken photo	Recognition time in seconds	Was recognition correct?
	0.094	yes
	0.12	no
	0.093	yes
	0.091	no
	0.11	yes
	0.096	no
	0.098	yes
	0.12	no
	0.099	yes
	0.097	no
	0.1	yes
	0.096	no

Figure 33: Test results for 10-class CNN (different lighting conditions and background visibility of input image).

.

Test for algorithm's scalability involved incorporating additional 10 classes (10 different beer labels) and retraining. Average time of a single beer label recognition, this time out of 20 classes, is 0.098 second. Figure 34 and 35 contain charts of training progression, which in the end took 1253.84 seconds and resulted in learning a total of 9574164 parameters.

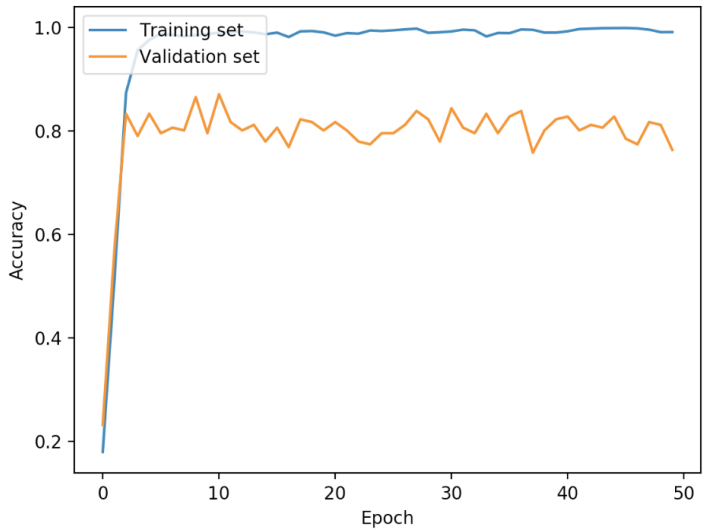

Figure 34: Accuracy of 20-class model during training. The best obtained value is 99.8% for training set and 87.6% for validation set.

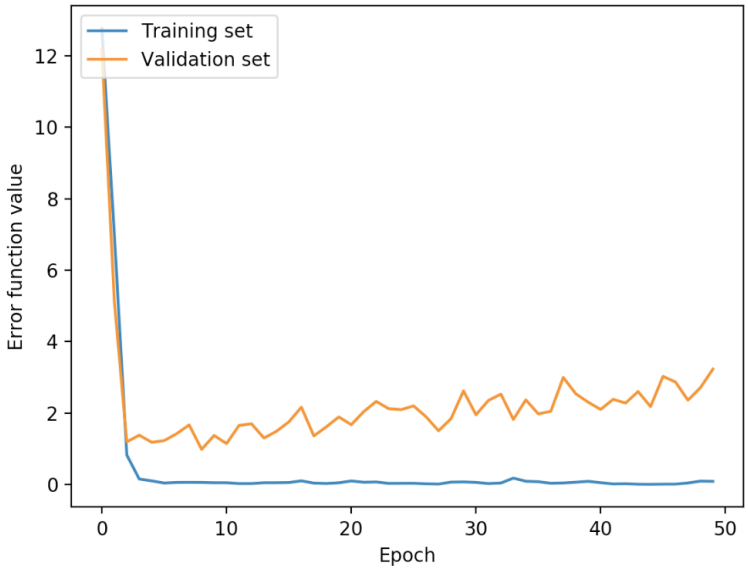

Figure 35: Categorical cross entropy error during training of 20-class CNN. The best obtained value is 0.017 for training set and 1.16 for validation set.

13 Conclusions

The evolution in customers' taste and their openness to local breweries resulted in a huge variety of beers available in popular Polish stores. Despite this fact, customers still find it cumbersome to choose which product to buy.

When faced with a mobile app with beer database, users consider it the most attractive to access beer details by making a photo of front beer label. Scanning bar code seems to be the least popular.

Artificial Intelligence (deep learning in particular) incorporated in an application can itself be a driving force behind rapid growth rate of users, following the likes of e. g. Shazam (mobile app used to recognize name of a song, given few seconds of its melody).

Recent increase in performance of GPUs and data available on the Internet provide a fertile ground for machine learning algorithms, such as neural networks.

Currently, the best way to train a neural network is by using a backpropagation algorithm. It is also utilized in CNNs, which are specialized in image recognition (they preserve spatial connections between pixels).

Thanks to ILSVRC competition hosted by ImageNet, many world-class CNN architectures emerged. It is particularly important because of the technique called transfer learning, which allows to adapt them to custom recognition tasks.

The more specific the task the higher possibility of manual data collection. However, instead of manually collecting thousands of images per class, one can artificially inflate training set by using data augmentation (applying a series of picture transformations).

Firebase Realtime Database, which contains data about beers, has currently very poor support for searching. The use of it excluded possibility of OCR implementation, which requires a solid partial query search engine.

Flask proved to be small and reliable framework for REST API creation. The prototype runs the server in the local network, however uploading it to Cloud (such as Google App Engine) is extremely easy and requires almost no additional code.

Implemented beer label classifier can achieve at least 87% accuracy under user-friendly conditions, with inference time not exceeding 0.2 second. Aggregation of additional beer labels does not significantly change accuracy nor recognition time. It can be concluded, that drastic increase in number of classes, which CNN is capable of identifying, will result in a classifier, which inference time won't surpass 1 second, while still maintaining around 90% success rate.

The accuracy of proposed solution does not depend on lighting conditions of taken image, however it is prone to failure when recognizing photos with a lot of noisy background.

There seems to be no popular and continuously maintained application solving beer label recognition problem using deep learning: Untappd app's developers implemented bar code scanning to accelerate beer database access. On the other hand, Vivino is using OCR but for wines only. Once popular NextGlass mobile application has not been updated since 2015.

Further development of this project could involve:

a) migrating to database, which supports better querying engine,

b) migrating from convolutional neural network to recently popular capsule networks for increased robustness (e. g. increased accuracy given a lot of background noise),

c) fine-tuning implemented CNN using different architecture and hyperparameters.

References

[1] *Repository of Android application used for beer label recognition.* https://github.com/matdziu/BeerAI-App, access: December 14, 2017.

[2] *Repository of REST server (Flask) used for receiving beer label image from client and sending predictions.* https://github.com/matdziu/BeerAI-Server, access: December 14, 2017.

[3] *Repository of scripts performing data augmentation.* https://github.com/matdziu/BeerAI-Data, access: December 14, 2017.

[4] *Repository of scripts performing inference and training of CNN.* https://github.com/matdziu/BeerAI-Model, access: December 14, 2017.

[5] *HSV color model definition.* https://en.wikipedia.org/wiki/HSL_and_HSV, access: November 18, 2017.

[6] *"Kotlin on Android. Now Official" blog post.* https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/, access: November 18, 2017.

[7] *NoSql definition.* https://en.wikipedia.org/wiki/NoSQL, access: November 18, 2017.

[8] *REST definition.* https://en.wikipedia.org/wiki/Representational_state_transfer, access: November 18, 2017.

[9] *Retrofit library repository.* https://github.com/square/retrofit, access: November 18, 2017.

[10] *RxJava library repository.* https://github.com/ReactiveX/RxJava, access: November 18, 2017.

[11] *Synset definition.* https://en.wikipedia.org/wiki/Synonym_ring, access: November 4, 2017.

[12] *Android framework guide.* https://developer.android.com/guide/index.html, access: October 24, 2017.

[13] *Firebase Realtime Database documentation.* https://firebase.google.com/docs/database/, access: October 24, 2017.

[14] *Flask documentation.* http://flask.pocoo.org/docs/0.12/, access: October 24, 2017.

[15] *Survey "Exploring new beers".* https://www.surveymonkey.com/results/SM-S32QWQ578/, access: October 24, 2017.

[16] *CPU vs GPU performance.* http://michaelgalloy.com/2013/06/11/cpu-vs-gpu-performance.html, access: October 26, 2017.

[17] *Chain rule definition.* https://en.wikipedia.org/wiki/Chain_rule, access: October 28, 2017.

[18] *RGB color model definition.* https://en.wikipedia.org/wiki/RGB_color_model, access: October 29, 2017.

[19] A. Canziani, A. Paszke, and E. Culurciello, *An analysis of deep neural network models for practical applications*, arXiv preprint arXiv:1605.07678, (2016).

[20] Cisco, *The Zettabyte Era: Trends and Analysis*, 2017.

[21] KPMG, *The alcoholic beverages market in Poland*, 2014.

[22] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object recognition with gradient-based learning*, Shape, contour and grouping in computer vision, (1999).

[23] D. G. Lowe, *Object recognition from local scale-invariant features*, in Computer vision, 1999. The proceedings of the seventh IEEE international conference on, vol. 2, Ieee, 1999, pp. 1150–1157.

[24] P. Norvig and S. J. Russell, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2010, pp. 5–16, 730.

[25] E. Ohbuchi, H. Hanaizumi, and L. A. Hock, *Barcode readers using the camera device in mobile phones*, in Cyberworlds, 2004 International Conference on, IEEE, 2004, pp. 260–265.

[26] E. Reis, *The lean startup*, New York: Crown Business, (2011).

[27] A. Weitz and A. Chaudhari, *Beer label recognition and classification.*

14 Necessary translations

Ta sekcja stanowi podsumowanie najważniejszych informacji dotyczących zaprezentowanej pracy dyplomowej.

1. **Definicja problemu.**
   Przyspieszenie dostępu użytkownika do bazy danych piwa przy wykorzystaniu wizji komputerowej i urządzenia mobilnego.
2. **Cel utylitarny.**
   Zaprojektowanie i implementacja systemu typu serwer-klient służącego do ropoznawania etykiet piwa.
3. **Cele poznawcze.**
   (a) Badanie wpływu architektury konwolucyjnej sieci neuronowej (KSN) i augmentacji danych na dokładność i czas trenowania rozpoznawarki.
   (b) Zyskanie głębszego zrozumienia problemu wizji komputerowej przy użyciu KSN.
   (c) Znalezienie minimalnej ilości iteracji treningowych i danych potrzebnych w celu osiągnięcia skuteczności rozpoznania nie mniejszej niż 85%.
4. **Zakres.**
   (a) Implementacja skryptu do augmentacji danych.
   (b) Implementacja skryptu do uzyskiwania tzw. bottleneck features i trenowania konwolucyjnej sieci neuronowej.
   (c) Implementacja serwera REST do wysyłania predykcji.
   (d) Impementacja aplikacji na Androida, pozwalającej na robienie zdjęć i wysyłanie ich na serwer.
5. **Teza.**
   Zaimplementowana konwolucyjna sieć neuronowa, rozpoznająca etykiety piwa, jest skalowalna i potrafi osiągnąć co najmniej 85% dokładności w warunkach przyjaznych dla użytkownika. Czas wnioskowania wspomnianej sieci nie przekracza 1 sekundy.

Spis rysunków - tłumaczenie