Contents

1 Problem definition

KPMG's report on "The alcoholic beverages market in Poland" claims, that in 2013, Poles spent 41.1 billion zlotys on alcohol, with beer being a large portion of that amount (47%) [9]. 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 [9]. 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 like Lidl and Carrefour.

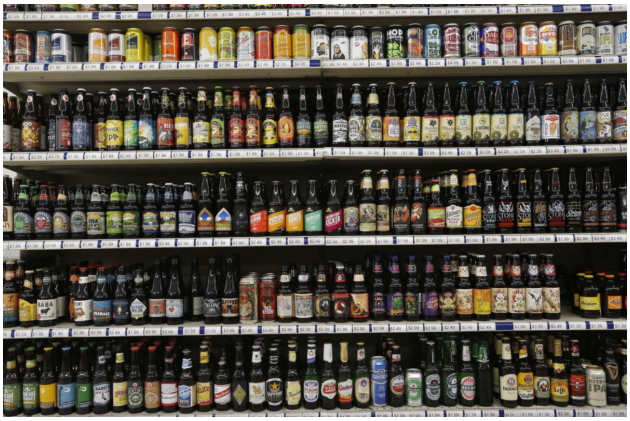

Figure 1: A large beer shelf. Source: `http://trib.com`.

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?".

2 Solution overview

To further investigate the problem articulated in Chapter 1, I conducted a survey [4]. Its main question, that I want to focus on in this section, 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 my personal experience, I want to formulate a solution to the problem, stated in Chapter 1, as a design and implementation of server-client recognition system for beer labels.

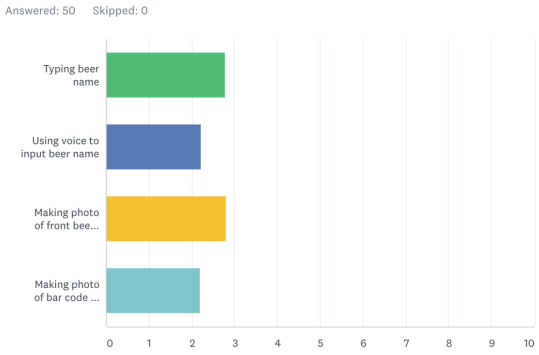

Figure 2: Weighted average of respondents' answers (chart). For choosing an option as first - weight: 4, for choosing an option as second - weight: 3 etc. Obtained sum is then divided by total number of respondents (here: 50).

	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.

The goal of this section is to give a reader, in a few short points, an overall idea of the recognition system before theoretical introduction. Chapter 9 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 [1] application.
2. Android application sends the image to the Flask [3] 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 [2].
5. Based on the key (encoded label name) all information about given beer is sent to Android application and displayed to the user.

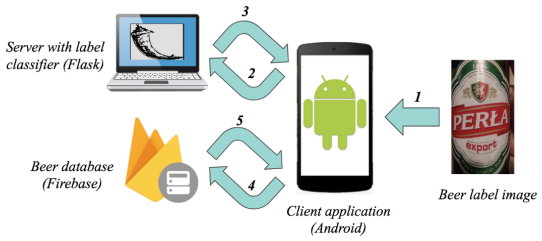

Figure 4: Server-client recognition system for beer labels with functional principle (consecutive steps enumerated).

3 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 did a remarkable job of listing major disciplines, that contributed to the growth of AI [11], 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).

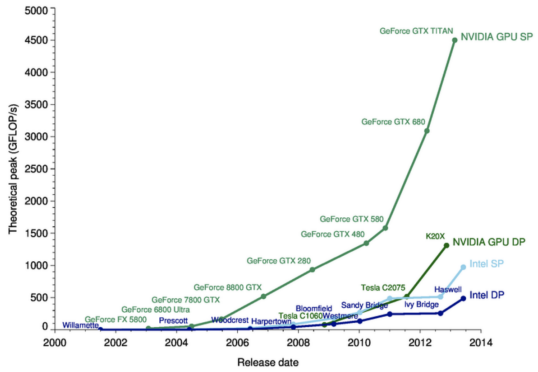

Figure 5: Performance of NVIDIA GPUs and Intel CPUs measured in GFLOP/s [5].

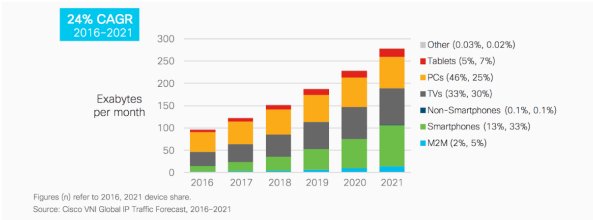

Figure 6: Global IP traffic by devices [8] with 24% compound annual growth rate (2016-2021).

I want to conclude this chapter with important distinction 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.

4 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.

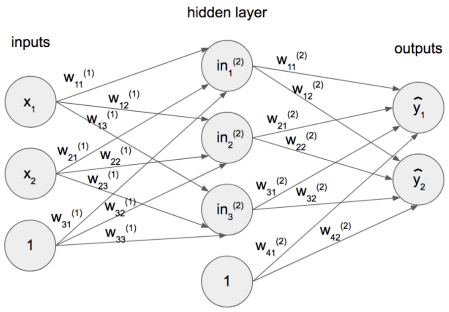

Figure 7: A simple neural network with one hidden layer (a layer between inputs and outputs).

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

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

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

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

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

"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." [11]

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 I want to introduce 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).

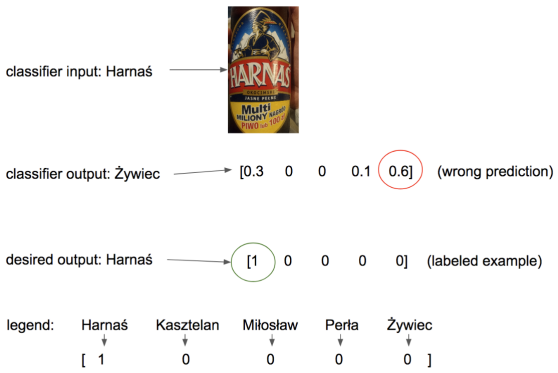

Figure 8: Exemplary classifier input, output and desired (one-hot encoded) output with legend.

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

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 I'll be introducing 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

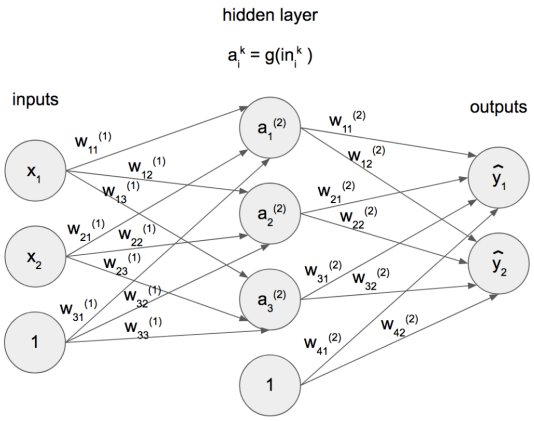

Figure 9: Updated model of simple neural network from Figure 7. g is a sigmoid activation function. Outputs are often normalized using softmax.

functions.

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

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

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, I performed, so called, forward pass, which is simply a mathematical inference of outputs, given inputs (Equation 10).

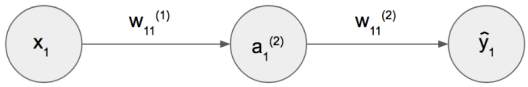

Figure 10: A simple neural network, with sigmoid activation function g (Equation 7), i. e. $a_1^{(2)} = g(w_{11}^{(1)} x_1)$.

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

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 [6]. 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:

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

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

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

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

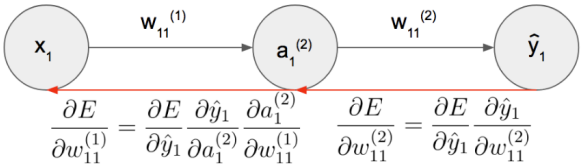

Figure 11: Outline of a single step of backpropagation with chain rule, where $E = -y_1 log(\hat{y_1})$. Value y_1 is a desired binary output from training set.

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

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

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).

5 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.

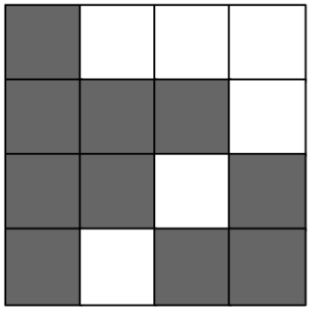

Figure 12: Grayscale image of number 7 (4 x 4 pixels resolution).

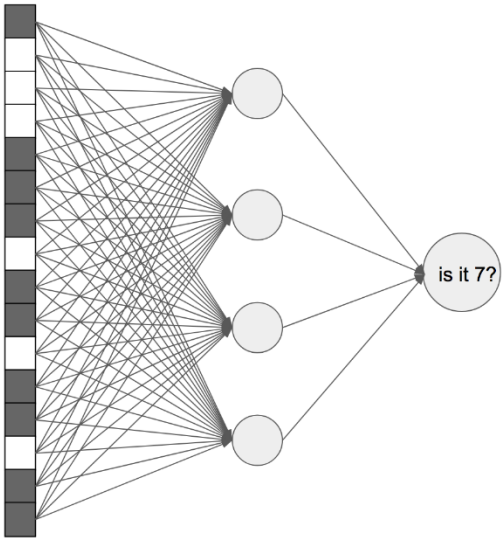

Figure 13: Regular neural network recognizing number 7 in image input from Figure 12, which is deconstructed (flattened) into vector of size 16.

The standard reference for ConvNets is *"Object Recognition with Gradient-Based Learning"* by LeCun et al. (1999) [10]. This section attempts to explain basic building blocks of the system designed in that paper, called LeNet-5 Figure 17. Since I've already mentioned about fully connected

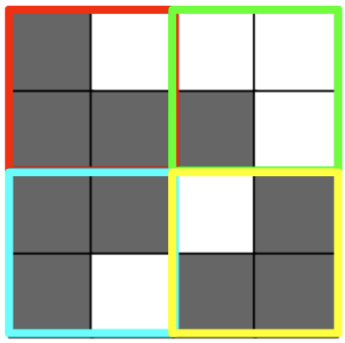

Figure 14: A number from Figure 12 divided into four pixel regions.

layers (standard NN layers) in the previous chapter and about flattening a matrix in Figure 13, I will focus on:

1) convolutional layer

2) subsampling layer

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 [7]). 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).

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).
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.

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

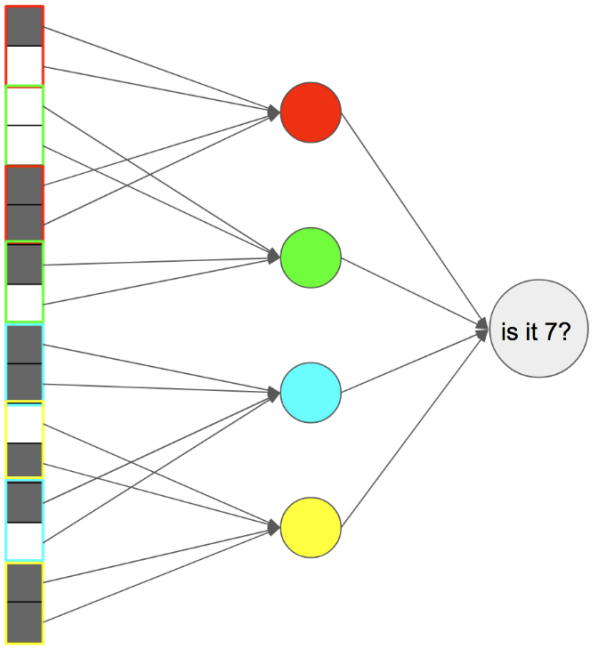

Figure 15: Local connectivity of inputs from Figure 14 to hidden nodes (vector form).

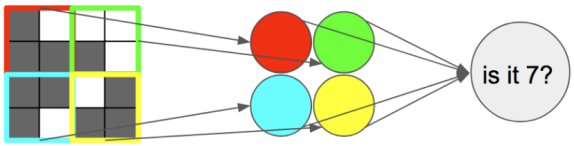

Figure 16: Local connectivity of inputs from Figure 14 to hidden nodes (matrix form).

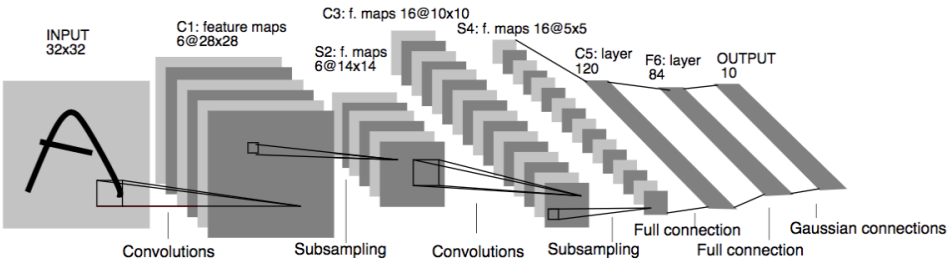

Figure 17: LeNet-5 CNN [10] used for handwritten digit recognition.

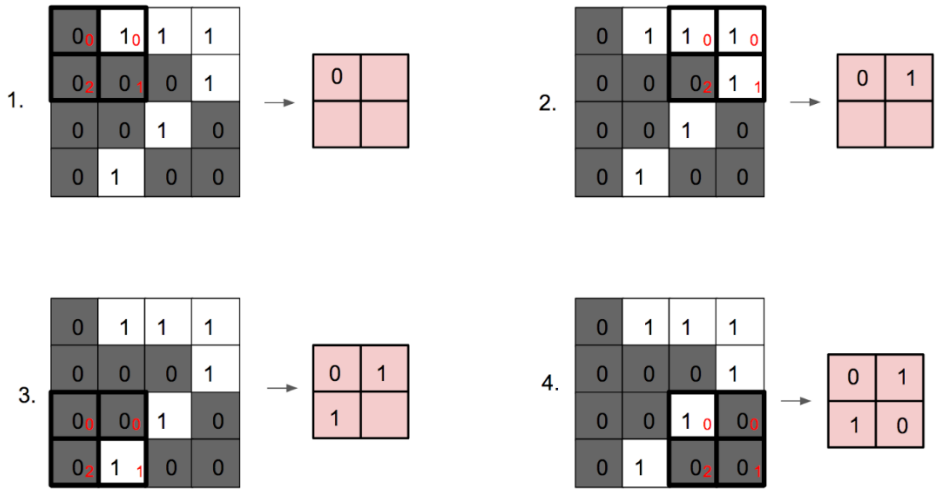

Figure 18: Consecutive steps of convolution using filter of size 2x2 with stride 2 (coefficients visible in red in the bottom right corner of a filter). The pink matrix is called a feature map and in this case represents a convolutional layer.

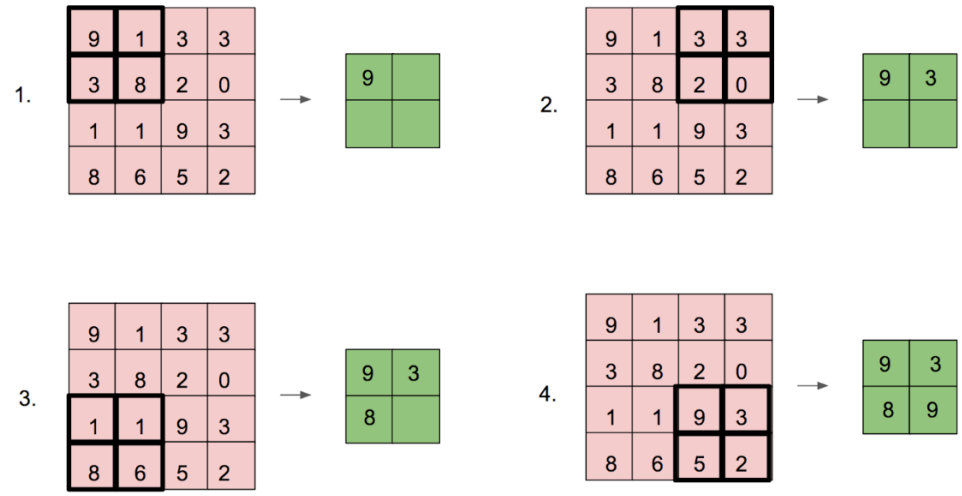

Figure 19: Consecutive steps of max pooling with window of size 2x2 and stride 2. Green matrix is a reduced tensor.

6 Comparison of world-class CNN architectures

7 Transfer learning

8 Data augmentation

9 Solution walkthrough

10 Testing

11 Related solutions

12 Conclusions

References

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

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

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

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

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

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

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

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

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

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

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