STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIV**, Number 2, 2019
DOI: 10.24193/subbi.2019.2.02

# ROAD CONDITION CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

GEORGE-BOGDAN MACA

Abstract. *Autonomous driving is an increasingly important theme nowadays. One of the reasons behind this is the evolution of hardware components in the last years, which made possible both research and implementation of much more complex deep learning techniques. An interesting direction in the vast field of autonomous driving is the discrimination of the condition of the road, with respect to weather. This paper presents a supervised learning based approach to road condition classification. Specifically, we take advantage of the power of Convolutional Neural Networks (CNNs) in the context of image classification. We describe several CNN architectures that use state of the art deep learning techniques and compare their performance. In addition to the simple CNN-based learners, we propose a CNN-based ensemble learner able of a better predictive performance compared to the single models.*

## 1. Introduction

In this paper we refer to *road condition* as the state of the road in terms of weather (i.e. if the road is bare, or covered by water or snow). By the *classification* of road condition we refer to the task of deciding what is the state of the road from a given image. It can be argued that this task can be quite complex, since there might arise situations difficult to distinguish even by the human brain. These are the cases that fall somewhere at the *limit* between two classes, when it is not obvious to which of those classes does the image belong. Additionally, since the data comes from real traffic scenes, the images might contain *noise* in the form of other objects present in the scene, that can obturate the view (e.g. cars, pedestrians, etc.).

Why do we need road condition classification? The concept of *self-driving cars* has been around for some years. Nowadays it is becoming increasingly

Received by the editors: June 25, 2018.

2000 *Mathematics Subject Classification.* 68T05,91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial intelligence**]: Learning – *Concept learning*.

*Key words and phrases.* Autonomous driving, Road condition classification, Supervised learning, Convolutional Neural Networks, Ensemble learner.

popular and we can already see it in real world scenarios. In the context of *driver assistance*, or potentially even *autonomous driving*, the braking system of a vehicle can be improved. Specifically, the parameters of the ESP (electronic stability program) system can be adjusted depending on the condition of the road. Another scenario, especially helpful for 2-wheeled vehicles, would be sending warnings to the driver in case of dangerous road state, so that he can reduce the speed in time. For example, the road could be wet and thus slippery, even though it is not obvious from the condition of the weather at that particular moment.

However, the necessity for road condition classification was first introduced not in the context of autonomous driving, but rather in the not so "luxurious" task of dealing with *extreme weather conditions* (especially during winter). In northern states like Norway, Sweden, Canada, etc. winters can be very harsh and usually have very bad effects on the roads and the traffic, implicitly. Therefore, these situations must be taken care of very responsibly and efficiently (by "efficient" we mean both fast and, if possible, with the smallest consumption of resources).

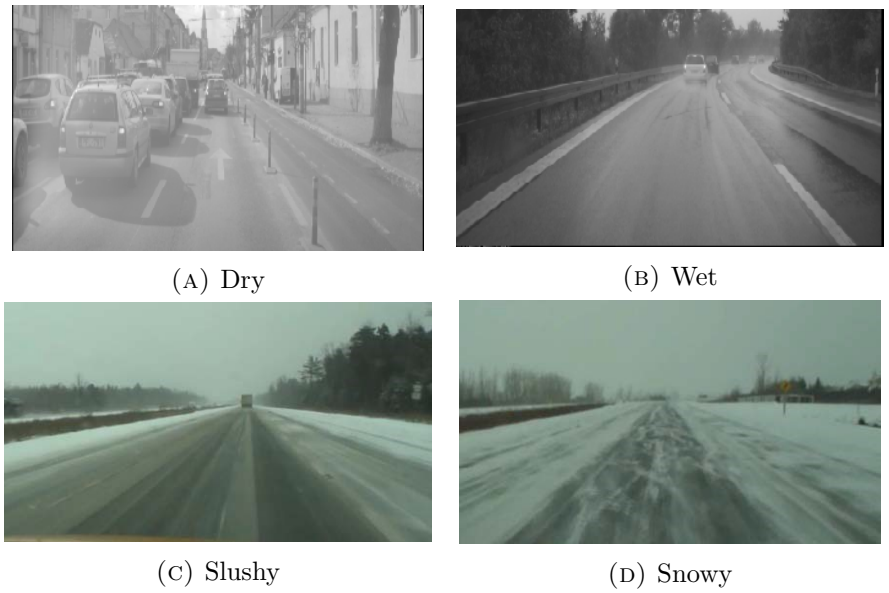
Currently, monitoring of winter road conditions is mainly done either using road weather information systems (RWIS) installed at fixed positions or through visual observation and manual recording by maintenance personnel. The former is limited in spatial coverage while the latter is limited in repeatability, objectivity and details [16]. Therefore, the need of automatic classification of the state of the road in a particular place (e.g. in one of the following classes: dry, wet, snowy, slushy, etc. - Figure 1) came up. The classification will be done *automatically* using *machine learning* techniques. More specifically, a direct approach of the form:

$$input - classifier - label$$

will be used, rather than the classical approach based on feature extraction:

$$input - features - classifier - label$$

In this problem we approach the task of classifying real world traffic scenes. The data consists of images obtained from video cameras mounted on cars. During the training process, each image (frame) is labeled with one of three classes (dry, wet, snow). The choice of output classes was conditioned by the state of our dataset, which was constructed and labeled based on images from these particular classes, that we were able to gather. Since the images are labeled, a *supervised learning* algorithm is more suited then an unsupervised one, mainly because the supervised approach would be more robust and certain of success. We have therefore chosen to use an approach based on *Convolutional Neural Networks* [12, 10].



(A) Dry (B) Wet

(C) Slushy (D) Snowy

FIGURE 1. Possible states of the road.

In this paper we present a supervised learning based approach to road condition classification, specifically using Convolutional Neural Networks. To the best of our knowledge regarding the state of the literature, we introduce a novel approach on this task, mainly due to the fact that we propose original architectures. There is also a very recent similar work [15] in which CNNs are used for Road Condition Classification. However, the authors of the paper [15] use state of the art CNN architectures [5, 21] pretrained on the ImageNet dataset [2]. Our works also differ in terms of chosen output classes and the conditions in which the images are recorded. The main contributions of this paper come from the fact that we propose multiple CNN architectures, built using various techniques. Also, we combine some of these architectures and obtain different models, including an ensemble of three learners. The models are evaluated experimentally and the obtained results are discussed.

The rest of the paper is structured as follows. In Section 2 we present another approaches to the same problem, found in literature. Section 3 offers a detailed explanation of our supervised learning approach, as well as a presentation of the architectures used. In Section 4 we present the performance of each of the proposed models. The results shown are then discussed in Section 5. Section 6 ends the paper with conclusions and potential improvements.

## 2. Related work

In this section we present a few approaches that are related to our work in different aspects. Such aspects are: the Machine Learning algorithms used, the information used as input (obtained from sensors or images), the type of input information (raw or processed). We focus on scientific works dedicated to road condition classification in scenarios similar to the ones available in our dataset.

2.1. **Friction measurement.** A solution rather related to physics, based on continuous friction measurement (CFM), was proposed in [3]. Friction is a measure of the resistive force to movement between the tires of a vehicle and the road surface and thus it represents an accurate approximation of the quality of driving on that road. In the study, they used the idea that the value of friction is inversely proportional with the quantity of snow on the road.

Friction measurements on a section of Highway 417 in eastern Ontario were obtained using a special equipment called Traction Watcher One (TWO). In addition to the sole friction measurement, the authors used some new parameters obtained from probability density pattern (e.g. variance and skewness) and from spectral density pattern of CFM. For this problem, 6 different classes were used: bare dry (Type 0), bare wet (Type 1), thin snow cover (Type 2), slushy snow cover (Type 3), partially snow covered (Type 4) and mostly snow covered (Type 5). After observing that some pairs of their chosen classes were very difficult to distinguish, the authors of the article decided to use a multilayer nested structure as their model. This architecture comprised five binary logistic regression models.

| **Split number** | **Validation accuracy** |
|---|---|
| 1 | 0.9383 |
| 2 | 0.9405 |
| 3 | 0.7957 |
| 4 | 0.8876 |
| 5 | 0.58063 |

Table 1. Results of the CFM approach. [3]

The results obtained in this paper, on the validation data set, are summarized in Table 1. It shows the performance of the five logistic regression models in turn. For each one of them, the authors decided what features to use and how to combine them, from the following list: *F* - friction, *Std* - standard deviation, *Skew* - skewness, *HighFreq* - high frequency power of CFM, *LowFreq* - low frequency power of CFM.

2.2. **Neural Networks approach.** Another approach to road condition classification is described in [11]. It is an image based approach that uses Neural Networks which receive as input feature vectors, carefully chosen from images. The purpose is to classify images in one of the following 5 classes: dry, wet, snowy, icy and snowy with tracks. Their motivation is to supplement already existing measurements (such as wind speed and temperature) with new information obtained solely from images. Also, the stage is set for this approach, in the sense that some measurement stations are already equipped with video cameras. The knowledge of the road state is used in order for the authorities to decide what kind of maintenance to perform and in what places.

After a careful analysis of what kind of information from an image is needed for this particular problem, the authors of the article end up with 6 features (e.g. mean gray level, standard deviation of the gray level, ratio of the standard deviation of the red image to the standard deviation of the blue image, etc.).

One of the downside of the proposed solution is the dimensionality of the architecture. More specifically, the neural network used had no more than 9 neurons (3 or 4 on the input layer; 3, 4 or 5 on the single hidden layer and one on the output layer – depending on the architecture and feature combination chosen). This is understandable, due to the fact that the dataset was very small – 69 RGB images in total . Thus, if the neural network had been bigger, it would have been very prone to overfitting. Mainly because of the two unfavorable aspects discussed above, the results obtained also make room for improvement: their best model and combination of features produces about 50% accuracy. However, as the authors of the article noted, there are several possibilities of improvement in this direction.

In addition to the work presented above, the authors of [15] introduce another approach based on Neural Networks, specifically CNNs. However, they use ResNet [5] and Inception [21] CNNs pretrained on the ImageNet dataset [2]. Also the output classes addressed in their problem are different from the ones used for our images.

2.3. **SVM-based approach.** In order to address the limitations of the currently existing methods of monitoring road surface condition (either using RWIS, or maintenance personnel) discussed in the first section, a new solution was proposed in [16]. It is based on using general purpose vehicles (like police cars or public transport) equipped with GPS systems and video cameras. Each image is GPS – tagged upon being taken and whenever there is an available internet connection, the tagged data is sent to a central server, where the actual computing takes place. This particular idea is very efficient in terms of time and space coverage.

As for the image classification task, a *support vector machine* (SVM) was used. The classes chosen for this problem are winter oriented: bare (dry), snow covered and track bare (with the center covered). Since the classification problem has more than two classes, the authors used a *one vs. all* multiclass technique. The data consisted of more than 500 RGB images (207 bare, 109 covered and 200 track bare). Each image was resized to 500x150 pixels before the feature extraction process. The dataset was split in the following way: 70% training images and 30% for testing purposes.

Results obtained with this approach are reasonably good, offering an average accuracy of 85% for the correct mapping of each of the three classes. These achievements are limited by various factors, like: the quality of the images, the amount of data and the illumination conditions, which tend to have a big influence in the color composition of an image.

There are several other approaches worth presenting, but many of them use methods based on features extracted from images (e.g. the one described in [17]). However, we want to direct our attention towards an approach that performs straightforward processing of data.

## 3. Proposed approach

3.1. **Theoretical background.** In this paper we present an approach to road condition classification based on *Convolutional Neural Networks* (CNNs) [12]. CNNs are the current state of the art in image classification tasks [20] and in the last decade have been used intensely for this kind of problems [10, 18, 21, 5]. Therefore, they represent an obvious direction to follow in the context of road condition classification. In addition, this technique has been also used in the context of autonomous driving, providing some good results (e.g. the approach described in [1], where the authors present an end-to-end learning method that maps an input image to a steering angle for the car).

CNNs are very similar to classical neural networks, in the sense that their ultimate purpose is to approximate a function, by training a set of weights. The main difference between them is that CNNs were created for the specific case, where the input to the network is an image. However, other types of inputs can also be fed to the network (e.g. audio, or even text) as long as they respect the *translational invariance* property (i.e. when it is not important exactly where something is located in an image). By making this assumption, convolutional networks can benefit from certain useful optimizations, which drastically reduce the otherwise huge number of parameters. The main advantage of a convolutional network over an ordinary neural network (when

working with images) is that they can obtain almost similar results with a much lower computational cost.

### 3.2. **Proposed architectures.**

3.2.1. *Overall architecture.* Before getting to the architectural details of the models used in this paper, we will first describe shortly each type of layer present in a convolutional neural network.

*Convolutional* layers are the core constructs of a CNN. Their behaviour can be understood as applying a filter operation on an image (i.e. each neuron on a convolutional layer computes a weighted sum of a group of pixels located at a particular position in the input volume). The parameters of a convolutional layer are the links between the neurons and the pixels they sum up.

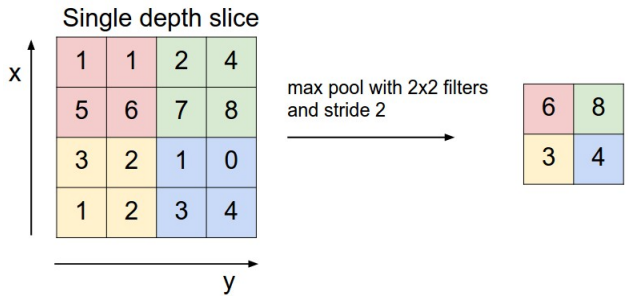



FIGURE 2. An example of max pooling [7].

*Pooling* layers are very similar to convolutional layers, the main difference being that pooling layers do not have learnable parameters. Their only purpose is to reduce the size of the input by performing *max* or *average* operations (no more weighted sums), as can be seen in Figure 2. *Dense* layers are nothing more than the *Fully connected* layers from the ordinary feed-forward neural networks. The neurons in this kind of layers are connected to all the neurons in the previous layer. It is a popular practice to use Dense layer towards the end of a CNN, just before the output layer.

A *Softmax* layer is used as the output layer of a CNN. It has the same number of neurons as the number of classes in the classification problem. The value in each of the neuron is interpreted as the normalized probability of the class corresponding to that neuron. The so called *softmax function* [8], which is able to compute those probabilities from the neurons on the previous layer, has the following form:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{1}$$

where: $j$ is the current output neuron; the sum over $k$ iterates over all the output neurons; $z_k$ is the value inside the $k^{th}$ output neuron after computing the weighted sum of its inputs.

3.2.2. *Proposed models.* In the following lines we will present the architectures used and discuss about the choices made. The first model we tried - let's call it **M1** - was inspired from the architectures used in [10] or [22]. More specifically, we started with bigger filters in the early layers (e.g. 7 x 7 in the first layer) and gradually reduced their dimension. This reduction happened after pooling layers or convolutional layers with strides of 2 x 2, so when the dimensions of the output volumes was reduced. In other words, this means that bigger filters would look at bigger "images" and this is intuitive since they can assimilate more information at a time, which seems to be desirable in the case of higher resolution images. Because the size of the filters decreases with the size of the image, we can say that their dimensions are directly proportional. However, it has been argued in the literature that this approach isn't actually practical (or at least, there exist better techniques) [18], but this will be discussed in more detail later, when we present another architecture we tried.

The second model used - call it **M2** - is a small adjustment to **M1** in terms of size. However, the overall architecture structure is similar. The two main differences between the models are the number of features maps at each layer (which is higher in the second model) and the number of parameters (which was also higher in **M2**). Although it wasn't highlighted in the description of **M1**, it is worth mentioning that both models contain a *Dense* layer at the end of the architecture, preceding the *Softmax* layer. The number of parameters in **M2** was drastically increased by using more neurons in that Dense layer.

In the third model used - call it **M3** - we incorporate and combine some more interesting concepts described in scientific papers. The authors of several papers in the literature [18, 21, 5] argue that the depth of a CNN is of central importance, especially for more complex tasks. This insight is actually intuitive, since the level of *abstraction* of the features extracted from a given input increases with the number of *consecutive (stacked)* layers in the network. Having in mind these assumptions, we decided to implement also a deeper architecture for the road condition classification problem. However, deep models come with several difficulties in the training process and we had

to make use of the ideas presented in the scientific papers in order to address these issues.

Firstly, we took advantage of the concept of *residual blocks* (see Figure 3), inspired from [5]. The intuition behind *residual connections* (or skip connections) is that they help the network to preserve information from the input, while going deeper through convolutional layers. Mathematically, the output of a residual block is of the form:

$$y = F(x) + x \tag{2}$$

where $x$ is the input of the block, $y$ is the output and $F(x)$ is the so called *residual mapping* (which is actually what an ordinary CNN would compute, given an input $x$). The " + " operator mentioned above is just an element-wise addition between the two operands involved, which means that their sizes must be equal along all the three dimensions (width, height and depth). This technique addresses the problem of *degradation* (i.e. the incapacity of very deep neural networks to converge even on the training set), as described in [5].

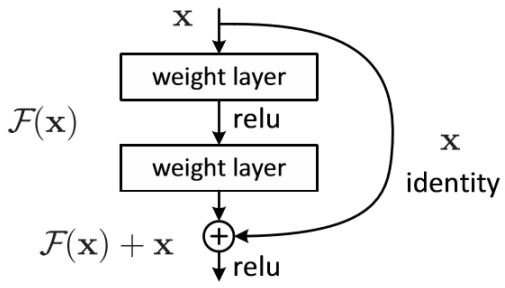



FIGURE 3. A building block of residual learning [5].

Secondly, we made extensive use of the of the *inception modules* introduced in [21], which can be visualized in Figure 4. The usage of this concept has at least two beneficial implications. The first advantage of inception modules is related to computational efficiency, since they introduce a parallelized manner of applying convolutions and pooling. The second insight is not as obvious as the first one and refers to the way information is extracted by the network. Specifically, at a particular layer in the network (i.e. given a fixed configuration of the input volume), information is extracted in more ways: by doing 1 x 1, 3 x 3 and 5 x 5 convolutions (the blue ones in Figure 4) and also a max-pooling (the red rectangle in the same image). This is not the case for ordinary CNNs, where only a single type of operation (e.g. 3 x 3 convolution, or max pooling,

etc) can be performed on **exactly** the same input (i.e. exactly the same values).

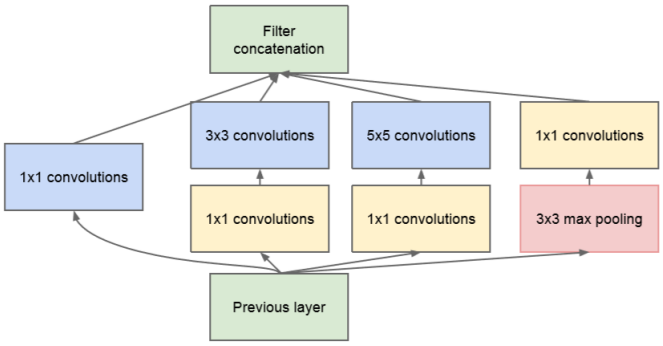



FIGURE 4. Inception module with dimension reductions [21].

The 1 x 1 convolutions from the yellow rectangles in Figure 4 are used for *dimension reduction*, as the authors of [21] argue. They say that performing convolutions with big filter sizes (e.g. 5 x 5, 7 x 7) is computationally expensive, so it would be better if they are performed on a small amount of feature maps in the input volume. Since the number of feature maps is given by the output of the previous layer and cannot be chosen arbitrarily, it was necessary to find a way to control that number. This is done by applying 1 x 1 convolutions on the input feature maps and choosing the desired number of output feature maps. In the case of 1 x 1 yellow convolution following the max pooling, its purpose is to control the number of feature maps that are concatenated with the convolutions; that number would otherwise be the same as the number of input feature maps, which would get higher and higher while going deeper through the network. It is also worth mentioning that in order to concatenate two or more volumes of feature maps, their spatial dimensions (width and height) must have the same size. So, these 1 x 1 convolutions in the yellow rectangles are used only for the reduction of the *depth* dimension, the other two dimensions remaining intact.

Last but not least, another interesting insight was gathered from [18]. As we mentioned earlier, we will describe another approach to choose the size of the kernels at each layer, rather than having big kernels in the first layers of the network and smaller ones towards the end. The authors of [18] mention the fact that two consecutive 3 x 3 convolutional layers perceive the same amount of information as a single 5 x 5 convolutional layer (a 7 x 7 layer is equivalent with three stacked 3 x 3 layers and so on). This implies that it is sufficient to use only filters of size (at most) 3 x 3 and this would have the

following advantages: reduce the computational cost induced by larger kernels and obtain CNN architectures that are deeper (which is desirable, as discussed earlier).

By combining the three ideas mentioned above (residual blocks [5], inception modules [21] and 3 x 3 kernels [18]) we end up with a network architecture made of building blocks looking like the one shown in Figure 5. There, one can observe the following aspects:

**1** Only convolutions with filter sizes of at most 3 x 3 are used.
**2** The parallelized block has also the role of the *residual mapping* $F(x)$ presented in Figure 3.
**3** In the parallelized block, the following convolutions are performed, from left to right, on exactly the same input: a 1 x 1 convolution, a 3 x 3 convolution and two stacked 3 x 3 convolutions (equivalent to a single
5 x 5 convolution).
**4** After the concatenation of the three branches, a 1 x 1 convolution is used in order to obtain a volume with the same number of feature maps as in the input, so that the element-wise addition between the input $x$ and the residual $F(x)$ should make sense.

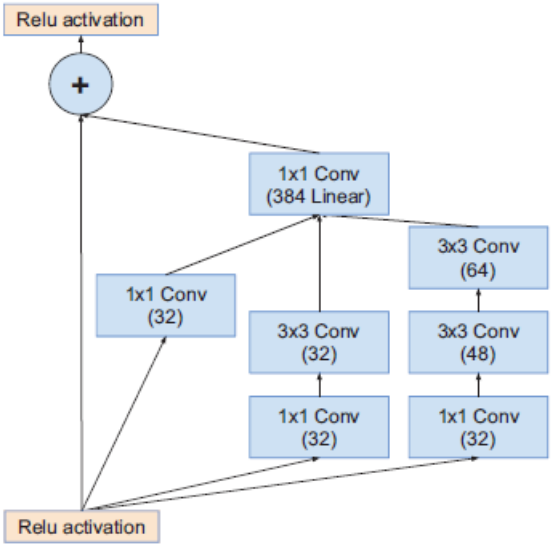



FIGURE 5. Example of Inception-ResNet building block [20].

An approach that combined these ideas is also presented in [20]. There, the authors also introduce the concept of *scaling* with respect to the addition

between the input and the residual mapping. They observed that computing a weighted sum that favours the input more would provide better results (at least in terms of convergence speed, if not actual accuracy). Thus, the " + " operator visible in Figure 5 would reduce to the following formula:

$$y = x + \gamma * F(x) \tag{3}$$

where $\gamma$ is the scale parameter that weighs the value of the residual and it should take values between 0.1 and 0.3, as the authors of [20] argue. We have also conducted some attempts to test the **M3** model with different values and combinations for the scale parameter.

A highlight of the main aspects which are different in the three models proposed is illustrated in Table 2. The last column in the table refers to the number of neurons on the Dense layer that precedes the Softmax layer (**M3** doesn't have such a layer). It can be seen that, even though **M3** is a much deeper architecture than **M2**, their number of parameters is comparable (**M3** has even fewer parameters). The reason behind this is the lack of Dense layers, which is affordable for **M3**, but not also for **M2**.

| **Model** | **Total number of layers** | **Number of conv. layers** | **Number of params.** | **Neurons on dense layer** |
|---|---|---|---|---|
| **M1** | 16 | 3 | 608,659 | 64 |
| **M2** | 21 | 5 | 840,986 | 256 |
| **M3** | 86 | 28 | 812,298 | - |

TABLE 2. Comparison between the architectures used.

Another approach we tried was to create a separate network for each one of the classes involved in the problem and make only binary predictions. Thus, each network would solve a classification problem having only two classes: positive and negative (e.g. the *"dry-CNN"* would answer to the following question: "Is the road dry?" with either "yes" or "no"). The idea behind this approach was to obtain a specialized network for each of the possible road condition categories, rather than a more general network. In the end, this ensemble of networks would provide its answer in the following way: given an input image, feed it through each of the specialized networks and obtain the scores $s_1, s_2, \ldots, s_N$ ($N$ being the number of classes), which are vectors with two elements representing the probabilities of the negative and positive classes respectively (these values sum to 1, according to *Softmax*); then take

maximum of the values of the positive classes from these vectors: $\max_{1 \leq k \leq N}(s_k[p])$ (where $p$ represents the position where the probability for the positive class is stored in the score vectors); finally, conclude that the image introduced is of the type of the network that produced the maximum positive probability.

3.2.3. *Learning methodology - details.* For the learning process we use a gradient descent algorithm, specifically *Adam* [9]. Also, *relu* [13] activation functions are used after every convolutional layer, in every model.

In order to prevent overfitting we use the simple and effective concept of a *Dropout layer*, introduced in [19]. The intuition behind this idea is that some random neurons are ignored (eliminated, *dropped out*) during training so that they are not able to influence the process and especially their neighbour neurons. In this way, the neighbours that would normally depend on this neurons (which are now dropped out) are obliged to deal with the situation on their own and thus learn a more general context. This approach reduces the "co-adaptation" between neurons, as mentioned by the authors of the paper.

In order to address the famous problem *vanishing and* ***exploding*** *gradients* [4], a technique called *Batch Normalization* [6] is used. However, the idea was initially introduced in order to deal with *covariate shift* (i.e. the distribution of the inputs changes from layer to layer). Batch Normalization is applied at a particular layer and performs a *standard deviation* (stdev) normalization, thus disallowing the values of the parameters to reach undesirably high values. The term *"Batch"* from the name of the method refers to the fact that normalization is applied to a group of input data at a time and not to each training example separately. The authors of [6] argue that this has two advantages: the gradient of the cost function over a batch of inputs estimates better the gradient over the entire data set; and normalizing one batch at a time is more computationally efficient due to the possibility of parallelization.

3.3. **Evaluation metrics.** As far as the evaluation process is concerned, two possible metrics are useful in the context of image classification (or mapping an input to a label in general): the *loss* and the *accuracy*. The loss value is more relevant from the machine learning perspective, while the accuracy is more intuitive and easier to visualize for users. Our architectures use the *log-likelihood cost* [8, 14], since it comes together with the Softmax classifiers. The value of the loss is computed with the following formula:

$$L = -\log\left(\frac{e^{s_y}}{\sum_k e^{s_k}}\right) \tag{4}$$

where $s_k$ is the $k^{th}$ element of the vector of scores (class probabilities), $y$ is the index of the correct label for the given input in the vector of scores $s$ and $s_y$ is the probability value of the correct class, outputted by the network. This result follows from the more general formula of the cross-entropy between two probabilistic distributions:

$$H(p, q) = \sum_x p(x) \log(q(x)) \tag{5}$$

where p is the "true" distribution, where the probability is concentrated on the correct class and q is the distribution of the estimated class probabilities (computed by the softmax function) [8]. The intuition behind the cross-entropy loss is that it quantifies exactly how far an estimated distribution outputted by the network is from the ideal case when the probability for the correct class is 1 and all the others are 0 (so, the farther the distributions are from each other, the greater the loss). In contrast, the accuracy measurement is much less realistic, since it just verifies if the maximum probability obtained corresponds with the actual correct class. So, for example, in a configuration with three classes, where the correct class for a particular example is class #3, an output with the following values: [0.33, 0.33, 0.34] would be classified as correct; this is far from ideal. However, the cross-entropy loss would heavily penalize this output, since it is not close at all to the correct output of: [0, 0, 1].

## 4. Numerical results

4.1. **The Dataset.** The classification problem we approached consists of three classes: *dry*, *wet* and *snow*. Our data set is split into three parts: training set, validation set and test set. Each of them is balanced in the sense that their images are equally spread between the three classes. The training set contains about 75000 images (with $\sim$25000 from each class) and the test and validation sets both contain approximately 15000 images (with $\sim$5000 from each class). The images are grayscale and have a 640 x 245 resolution.

The images from the dataset are obtained from video sequences filmed in real world traffic with a camera mounted on a car, which means that consecutive frames dumped from a sequence are very similar. Therefore, one additional aspect becomes relevant to the diversity of the input images: from how many different video sequences a particular bunch of images was obtained. This comparison would make sense only in a "per-class" context; a visual representation is shown in Table 3. The labeling was done per-sequence, which means that all the frames in a particular sequence have the same label. This aspect has some negative implications which will be discussed in Section 5.

| Dataset | Dry sequences | Wet sequences | Snowy sequences |
|---|---|---|---|
| **Training** | 82 | 52 | 12 |
| **Validation** | 11 | 5 | 2 |
| **Test** | 9 | 5 | 4 |

TABLE 3. Number of different sequences that lead to their per-class corresponding image sets (for example, the 25000 images labeled as "dry" from the training set were obtained from 82 different sequences, while the 5000 images labeled with "snow" from the validation set were obtain from only 2 sequences).

In order to train the specialized binary classifier models, we created three more datasets from the original one. The labels of these new datasets were adjusted in the following way: for each of the three networks, the corresponding class would have a *positive* label (e.g. value 1) and the other two classes would have a negative label (value 0). After this change in the labels, the negative class images became two times more than the positive ones. Thus, the dataset had to be modified such that the number the two numbers became equal. This was done by sampling half of the images with a negative label.

4.2. **Results.** For evaluation purposes, let's call by **M4** the model obtained from the ensemble of three specialized models (one for each class), each doing a binary classification. All the three models have the same architecture as **M3**, the only difference being that their softmax layers contain two neurons instead of three. Table 4 summarizes the results obtained by each of the four models discussed on both the validation and test sets. The accuracy metric is used in this illustration for simplicity and visibility. However, the model parameters have been chosen in terms of loss actually, even though there could have been configurations with higher accuracy (but also higher loss - which is not desirable).

In addition to the accuracy comparison of the four models, Table 5 illustrates the results provided by the three specialized binary classifiers in turn. We are going to provide next some technical data regarding our models. The training time ranges between 40 minutes for **M2** to 70 minutes for **M3**, while the validation and testing times range between 2 minutes and 5 minutes respectively, for the same two models. In addition, a prediction with **M2** yields approximately 7 frames per second (fps), while a prediction done using **M4** is much slower, providing only $\sim$2 fps.

| Model | Validation accuracy | Test accuracy |
|---|---|---|
| **M1** | 81.16% | $76.61 \pm 0.653\%$ |
| **M2** | 92.41% | $84.76 \pm 0.555\%$ |
| **M3** | 86.68% | $82.55 \pm 0.586\%$ |
| **M4** | 95.88% | $87.70 \pm 0.507\%$ |

TABLE 4. Accuracy comparison of the four discussed models in the 3-way classification problem. For the test accuracy metric we expressed the values using a *Confidence Interval* (CI) of 95%

| Model | Validation accuracy | Test accuracy |
|---|---|---|
| **dry-CNN** | 90.25% | 87.81% |
| **wet-CNN** | 96.37% | 89.94% |
| **snow-CNN** | 99.00% | 90.01% |

TABLE 5. Performance of the three specialized models in the 2-way classification problem.

## 5. Discussions and challenges

The **M1** model isn't particularly efficient, mainly due to its reduced size. Nonetheless, exactly this issue provided us with an interesting insight concerning the size of a CNN. Specifically, the **M1** model was able to converge on the training set, but it performs poorly on the validation and test sets. Our first intuition led us to believe it was an overfitting scenario caused by a high number of parameters. However, we went on to realize it was actually the reduced size of the network that was causing problems. An explanation of this situation could be that the small network made its best effort to fit the training data but, due to its reduced size, it was restricted to finding only a local minimum, being unable to generalize. We managed to prove this insight in practice with our second model, **M2**, which only had more layers and more neurons on the Dense layer compared to **M1**, the overall architecture style being the same.

The second model, **M2**, achieved surprisingly good results despite being only a regular *sequential* model (i.e. it doesn't contain parallelized building blocks or skip connections). It even outperforms the much deeper **M3** architecture, as can be seen in table 4. The reason behind this is still under study, but we are inclined to believe that either the model wasn't trained for an enough amount of epochs, or the lack of the Dense layer is affecting the

performance (although this doesn't happen in the case of the 2-class models, which after all are also of type **M3**).

One can observe in Table 4 that the ensemble of learners (**M4**) provides the best results in terms of accuracy. The result is strengthened by the fact that the confidence interval for the accuracy of this model does not overlap with the CI of any other model. This situation would also be predictable by looking at the good results obtained by each of the three specialized models, available in Table 5. However after analyzing the test accuracy of the 2-class models, we would expect a test accuracy of around 89% for the ensemble of models (**M4)**, by taking the average value. This is not the case, since the actual test accuracy is $87.70 \pm 0.507\%$, as can be seen in Table 4. This happens because the *max* operation used to combine the outputs of the three models is not ideal.

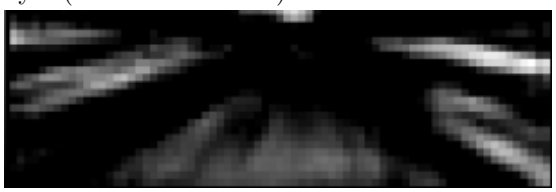
The main challenges encountered in our approach are related to the dataset. Firstly, some of the sequences contain images that lie at the limit between two classes. Such images are sometimes difficult to classify even by a human, not to mention an artificial neural network. Secondly, as discussed in Section 4.1, all the images obtained from a sequence share the same label. Obviously, some portions of the road in a sequence may not agree with the overall chosen label for that sequence and thus an inconsistency is introduced.

Other challenges we met are linked to the difficulty of understanding the behavior of a convolutional neural network. As also mentioned in Section 1, CNN is a direct approach that doesn't need manual feature extraction performed by the programmer. We can say that a CNN is able to decide by itself what features are needed in the learning process. In order to get a look inside the *black-box* of a CNN, we plotted the values of the neurons at different layers in the networks (see Figure 6). The white pixels in the subfigures 6b and 6c represent points of interest for the network. In this way one can get in insight regarding to where does a CNN "look" when it is fed with an input image.

We encountered difficulties in finding a relevant comparison between our results and the results obtained in the other approaches to road condition classification, presented in Section 2. There are multiple reasons behind this and we are going to briefly describe them. First of all, the datasets used for evaluation in other approaches from the literature are not published, so we couldn't evaluate our model on the same data. Even if the data was available, the problems approached are not identical, since the classes used for classification differ slightly in each paper. In addition, as far as we know there isn't any public benchmark containing data for the road condition classification problem. Moreover, the implementation details present in the papers from

(A) Input image.

(B) The plotted values of neurons of a feature map situated after the second convolutional layer (and its activation) of the **M2** model



(C) The plot of another feature map, situated after the third convolutional layer and its activation, in the same model

FIGURE 6. Looking inside a CNN

Section 2 are insufficient, so we were not able to reproduce the models and evaluate them on our data. Of course that we can analyze the accuracy values and compare them directly, but this is far from ideal and irrelevant.

## 6. Conclusions and future work

In this work, we presented a CNN-based road condition classification. We performed an in-depth analysis of several flavours of CNN-based single classifiers and we developed a CNN-based ensemble classifier that involves a bagging strategy with a max voting operator. We obtained a significant increase in the quality of classification (in terms of accuracy) by the developed ensemble

learning method. Our approach, albeit it is still in research phase, could potentially be used as a building block of an autonomous driving system, especially in terms of prediction efficiency.

After observing the initial success of our approach, we are convinced that *future work* should follow. Up to this point, the main improvement that we have in mind is related to the main challenge to our approach: the *dataset* (as we discussed in Section 5). The existence of images with uncertain labels isn't necessarily a strong estimate of the quality of the network, since some system might need to know only certain states of *dry, wet or snow*. Therefore we are planning to split the dataset into two, difficulty oriented sets: an easy one and a more difficult one. In this way we will be able to asses the performance of our architecture in a more realistic environment and also prove that the network performs bad mostly in uncertain conditions.

In addition to this, we also aim at improving the architectures, by trying more combinations of layers and hyperparameters with the purpose of increasing the classification quality even more. In order to obtain better results in terms of time and memory usage, we will try to adapt our CNN implementation to the specific properties of the board computers. Last but not least, we would like to consider a different method for splitting the dataset into train, validation and test sets. Currently the images are randomly split into the three datasets and we could use for example an algorithm like *K-Fold*.

## Acknowledgment

The technical support of Lucian Cristea, my mentor during internship, is highly appreciated.

## References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[3] F. Feng, L. Fu, and M. S. Perchanok. Comparison of alternative models for road surface condition classification. In *TRB Annual Meeting*, 2010.

[4] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[7] A. Karpathy. Convolutional neural networks: Architectures, convolution and pooling layers. 2016. last access (july, 2018).
[8] A. Karpathy. Softmax classifier. 2016. last access (july, 2018).
[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
[11] A. Kuehnle and W. Burghout. Winter road condition recognition using video image classification. *Transportation Research Record: Journal of the Transportation Research Board*, (1627):29–33, 1998.
[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
[13] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
[14] M. A. Nielsen. Neural networks and deep learning. 2015. last access (july, 2018).
[15] M. Nolte, N. Kister, and M. Maurer. Assessment of deep convolutional neural networks for road surface classification. *arXiv preprint arXiv:1804.08872*, 2018.
[16] R. Omer and L. Fu. An automatic image recognition system for winter road surface condition classification. In *Intelligent transportation systems (itsc), 2010 13th international ieee conference on*, pages 1375–1379. IEEE, 2010.
[17] Y. Qian, E. J. Almazan, and J. H. Elder. Evaluating features and classifiers for road weather condition analysis. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 4403–4407. IEEE, 2016.
[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.
[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.
[22] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania

*Email address*: mgic1759@scs.ubbcluj.ro