AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE



Wydział Inżynierii Metali i Informatyki Przemysłowej



PRACA DYPLOMOWA MAGISTERSKA

pt.

„Application of deep learning approach for identification and classification of scale defects during hot forming process”

Imię i nazwisko dyplomanta: **Szymon Furman**

Kierunek studiów: **Informatyka Stosowana**

Specjalność:  **Systemy Informatyki Przemysłowej**

Nr albumu: **268943**

Promotor: dr hab. inż. Łukasz Rauch

Promotor z przemysłu: Luc Van de Putte, ArcelorMittal Poland

Recenzent: prof. dr. hab. inż. Jan Kusiak

Podpis dyplomanta: Podpis promotora:

Kraków 2018

„Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia

4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90,

poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do

autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega

grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej

karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy

utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo

publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub

nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211

ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r.

poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny

uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed

komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym

dalej "sądem koleżeńskim"”, oświadczam, że niniejszą pracę dyplomową

wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż

wymienione w pracy.”

Kraków, dnia …… Podpis dyplomanta…………….

**Table of Contents**

[1 Introduction 4](#_Toc1932670)

[1.1 Motivation 4](#_Toc1932671)

[1.2 Aim of Thesis 4](#_Toc1932672)

[1.3 Content of Thesis 5](#_Toc1932673)

[2 The problem of defects in hot-forming sheets 6](#_Toc1932674)

[2.1 Hot forming process 6](#_Toc1932675)

[2.2 The phenomenon of scale formation in the hot forming process 8](#_Toc1932676)

[2.3 Factors affecting the formation of scale 10](#_Toc1932677)

[2.4 Scale defect in the hot forming process 11](#_Toc1932678)

[3 Classification of defects and division of the scale 13](#_Toc1932679)

[3.1 Rolled-in primary scale 13](#_Toc1932680)

[3.2 Secondary Scale 14](#_Toc1932681)

[3.3 Rolled *“V”* scale 14](#_Toc1932682)

[3.4 Peeled roll scale 15](#_Toc1932683)

[3.5 Red scale / Tiger scale 16](#_Toc1932684)

[3.6 Heavy scale 17](#_Toc1932685)

[3.7 Bad descaling 17](#_Toc1932686)

[4 Methods of recognition and classification of images 19](#_Toc1932687)

[4.1 Minimal distance methods 19](#_Toc1932688)

[4.2 Approximation methods 21](#_Toc1932689)

[4.3 Probabilistic methods 21](#_Toc1932690)

[4.4 Potential methods 22](#_Toc1932691)

[4.5 Stochastic approximation 23](#_Toc1932692)

[4.6 Decision trees, DT 23](#_Toc1932693)

[4.7 Artificial neural network, ANN 25](#_Toc1932694)

[4.8 Support vector machine, SVM 27](#_Toc1932695)

[4.9 Deep Neural Networks 29](#_Toc1932696)

[5 Image classification system based on CNN 33](#_Toc1932697)

[5.1 Data pre-processing 34](#_Toc1932698)

[5.2 Prepared datasets 38](#_Toc1932699)

[5.3 Convolutional Neural Network 42](#_Toc1932700)

[6 Results of trainings 44](#_Toc1932701)

[7 Bibliography 45](#_Toc1932702)

# Introduction

## Motivation

The main requirements of Modern Industry is to create high quality materials, with minimization of costs. Unfortunately, this is very difficult or even impossible to obtain that, using classic methods of production.

Steel industry and hot forming process have same requirements. Direct customers, and next steps of production expect to get high quality products. It causes, that process require constant control quality of product. One of the factors is rating of surface quality. This is a difficult process, because of velocity of hot forming process, multiple different defects and large surface, where we can find defects. One of the known inspections is a visual inspection made by inspector, which observe samples of produced plate. However, technological process provide new, automated solutions. One of the solution is a ASIS (Automatic Surface Inspection System). Main tasks of ASIS is take photos during the process, detection local difference of contrast, which can be a defect, and recognize proper class. The output of ASIS is classified defects map of produced material. This kind of solution allows to limit time for inspections and control of surface. Additional advantage is possibility getting information about state of production.

## Aim of Thesis

Aim of the thesis is to create a classifier of defects which can detect, and classify defect, founded on picture taken during hot production process. Based on local differences of contrast, classifier can detect, and in next step classify the defect. There are multiple different of class. To obtain desirable class, there must be prepared set of classified defects. Based on this set, classifier studies the vision features of pictures, and create a model. This is a typical *blackbox* approach for end user, because it allows to detect defects without specialized knowledge.

As for now, there is many approaches to solve this problem. Many of them is related on artificial intelligence e.g. *decision trees, neural networks etc*. This thesis is focusing on using Neural Networks using Convolutional neural networks (CNN). Convolutional neural networks is a class of deep, feed-forward Artificial neural network that has successfully been applied to analysing visual imagery and classify it. Based on classified set of pictures, solution will try to learn network model. Proper learned model will detect and classify defect. Eventually this solution can be used during hot forming process. Properly trained network provide fast and exact method to detect and classify defects, which can help inspector.

## Content of Thesis

In second chapter, thesis describe hot forming process, current state of sheet metal defect inspection and approximate the theoretical basis phenomenon of scale formation in the hot-forming process. In the next chapter, are presented Classification of defects, and their division. Fourth chapter describes the methods of recognition and classification of defects. It is focused on methods related with machine learning and AI. In subsections of this chapter are described different methods which can be used to image recognition and classification. Next chapter describe application of deep learning to analysing and classification of pictures. It is focused on *deep learning* and Convolutional neural networks.

# The problem of defects in hot-forming sheets

## Hot forming process

Subject of this thesis is about apply methods of neural networks to classification surface defects of flat sheets. Produced sheets and strips in their section are rectangle, and there have much more width than height. In view of temperature of forming, we can distinguish hot and cold forming process. Thesis is focusing on hot forming process.

Depending on the form, flat steel are divided into: sheets and strips. The sheet is called product which is hot or cold forming, flat with freely formed edges, supplied in form of rectangle sheets with 600 mm width and higher. The strip is called flat product formed on hot or cold, which is rolled into a circle directly after the final operation of forming, etching or continues annealing. Flat steels are used to production of cars, household goods, packing, etc.

Plastic deformation of flat steel at ambient temperature makes their harder, stronger and their changes electrical properties. At the same time, when deformation value increase. In order for the steel to be deformed, without any breaks, they should be restored earlier plastic properties. The phenomenon of reconstruction of the cellular structure, which is leading to recovery of the plastic properties is called recrystallization. Temperature of recrystallization in the absolute scale is described by the following empirical formula:

where Tr is a temperature of recrystallization, and a Ttop is a temperature of melting. For a pure metals the value is equal to 0.4, and for the alloys with the solid structure the value is equals to 0.6.

Hot plastic processing is made above the temperature of recrystallization. Desired shape of sheet or strip is obtained by plastic deformation of the material, between rotating and cooperating rolls. Thesis deals with longitudinal rolling (*Figure 1a*), where material performs a translational movement, and rolls with the parallel axes rotate in opposite direction.

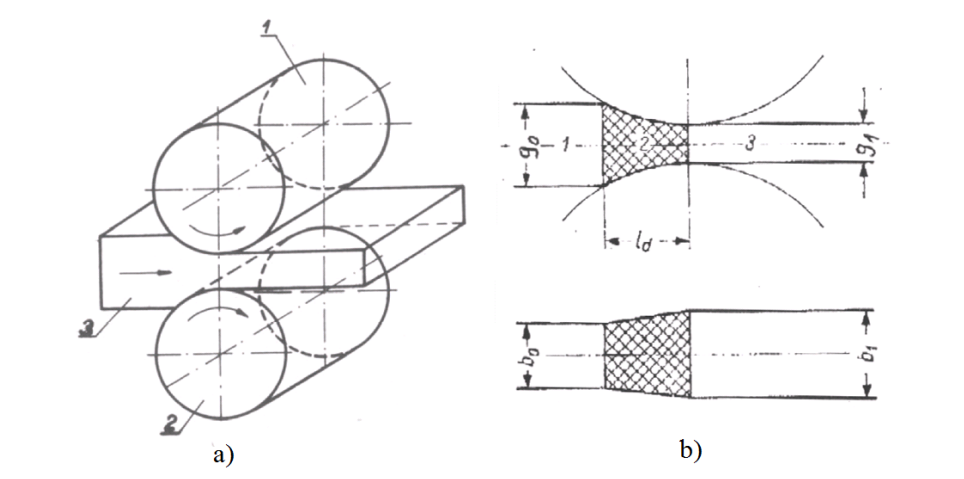


Figure 1: a) Longitudinal Rolling, b) Rolling mill

During of the rolling the cross-section is decreased, while the length is growing. Area, where the part of material is deformed is called rolling mill (*Figure 1b*). In the rolling mill occurs the reduction of thickness (from g0 to g1), and in the effect are changing: width, cross-section and length of sheet or strip.

Figure 2 shows the general concept of the layout of a typical hot rolling mill. The process starts from the loading of the ingot(1) into the furnace(2), which heats it to temperature around 1250 C. Ingots from the furnace will go to the milling line, which leads through the next technological steps. Initial mill(3), which job is initial reduction of a strip, can consists of a single reversing mill or several rolling mills in a similar arrangement. Some rolling mills use an intermediate band winding box (4) to reduce the length of the rolling line and equalization of temperature along the rolled strand. Installation of descaling (5) is responsible for removing scale from the surface of the strip. Group of finishing mills (6) gives the final thickness for the strip. Cooling section (7) sets the winding temperature. One coiler or group of coilers (8) forms the final product – circle of the tape.

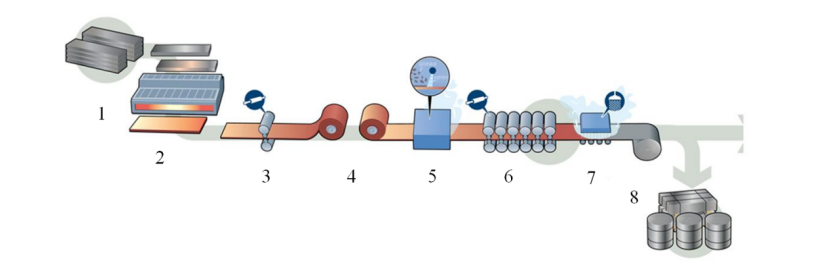


Figure 2: Diagram of the hot rolling mill

## The phenomenon of scale formation in the hot forming process

Scale defects are an effect of gas corrosion, when metallic material will be exposed to the environment with oxidizing properties in raised temperature. Reaction products are created, which can be volatile, liquid or solid in case of conditions and plastic composition. Solid parts of material are named scale defects.

The basic type of reaction which causes gas corrosion and forming of scale defects is chemical reaction of oxidation. In case of oxidation of pure divalent metal, it can be change as:  
where M is a metal, and X2 is a oxidation, for example: O2, S2, N2.

After a few seconds of reaction, the scale layer is higher than 10 µm. Their composition and construction differ depending on reaction conditions, reaction environments and time of oxidation. The chemical composition of the metal or alloy, where scale defect is precipitated has a significant influence. Commonly occurring scale defects are two-phase or multi-phase scale defects.

Beginning of reaction, causing formation of scale defect is characterized by the formation of thin scale. That scale is compact and adjacent to the ground (Figure 3a). If metal can create several persistent compounds with the oxidant, then scale defect will be multi-phase (Figure 3b). The thickness ratio of scale defect layers depends of temperature of the gas corrosion process. When the temperature growing, the thickness of direct layer adhering to material are growing to. Long oxidant process between metallic core and primary, compact scale defect layer causes creating porous inner layer. *Figures 3c and 3d* are showing one-phase and two-phase construction of scale defect at a later stage of its formation.

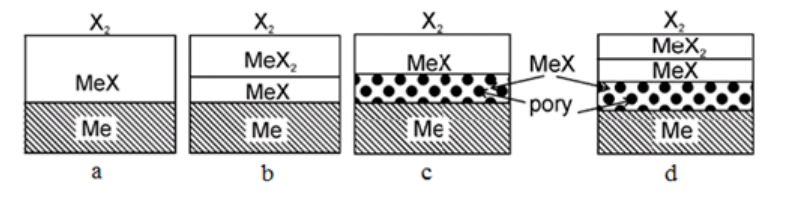
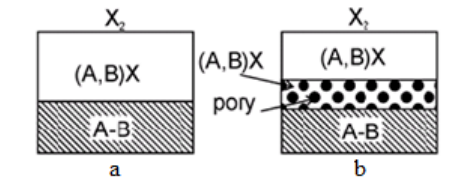


Figure 3: Formation of scale defect. a) one-phase in the initial stage of formation; b) two-phase in the initial phase of formation; c) one-phase at a later stage of formation; d) two-phase at a later stage of formation

In case of two-components alloys, where both alloys constituents (A and B) can create chemical compounds with oxidant X in reaction conditions, and assuming that the mutual solubility of the components AX and BX is unlimited, the one-phase scale defect will be created (*Figure 4*).

Figure 4: Scale formation for compounds AX and BX, which exhibit unrestricted mutual solubility



The other case of mentioned example is situation, when the compounds of two metals and oxidant do not exhibit mutual solubility and metals differ in their chemical affinity to the oxidant. In that case, homogeneous or heterophasic scale could be created. First scale in outside layer contains the only one compound (*Figure 5a*). It is an example of selective oxidation(Mrowec and Werber, 1982). Heterophasic scale(*Figure 5b*), in outside layer contains a mixture of compounds of both alloy components with an oxidant. This type of scale is formed, for example, on the alloys Fe – Cr in an environment containing oxygen or sulphur as the oxidant. In the case, where the AX and BX compounds have limited mutual solubility under the reaction conditions, it is possible to create two-layer scale with a continuous outside layer.

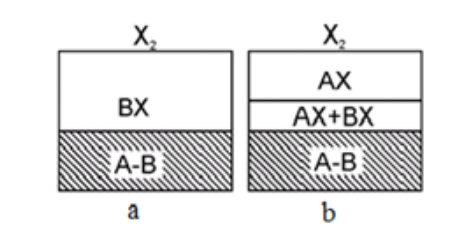


Figure 5: Scale: a) two-component in case of selective oxidation; b) heterophasic

In the last example, when the oxidizing component of the atmosphere can diffuse into the alloy and form a compound with a metal with higher chemical affinity (e.g. B), before the alloy constituents manage to diffuse into the surface, on metal A rich alloys, BX compounds will form underneath the layer of BX scale. The area of occurrence distributed BX phase in the surface layer of the alloy is called the internal oxidation zone. On metal-rich B alloys, the scale is formed, builded only from the BX phase, under which an internal oxidation zone may be formed(*Figure 6*)



Figure 6: Scheme of the structure of the scale and the internal oxidation zone in the case of the formation of a single-phase reaction product, when the chemical affinity of component B is higher than that of component A: a) rich in metal A; b) alloy rich in metal B

## Factors affecting the formation of scale

Formation of scale is depends mainly on the heating temperature, heating time, condition of metal surface, on which metal is formed, the chemical composition of gas, in the working space and the chemical composition of the steel. Scale may be compact and tight, then it is a protective layer, obstructing access of external gas to the metal surface, and thus will stop the oxidation process. The examples of metals, which can create compact scale in higher temperature are e.g. copper, chrome, zinc(Czermiński, 1978)

In the process of scale formation an important elements is the condition of metal surface, which has the effect of gas corrosion, and then the scale is formed. Acceleration of this process may be due to the privileged crystallographic orientation of the metal surface exposed to oxidizing atmospheres(Dobrzański, 2002).

The factor, which facilitates the process of gas corrosion is the mechanical machining of metal. When the internal stresses increase, there is an increase in network defects in the outer metal or alloy layer. As a result of local heating of metal, layers or oxides or hydroxides are formed.

The profile of the metal surface also has an effect on the speed of the descaling effect. If the surface are exposed to the effect of gas corrosion is high and has irregular shapes, then it favours the uneven formation of scale and the occurrence of cracks and micro-voids.

Character of the chemical reactions and the magnitude of corrosion damage is determined mainly by the chemical composition. To gases, which can cause an intense course of gas corrosion include: air, water vapour, carbon compounds e.g. mixtures of CO-CO2 and hydrocarbons, sulfur compounds, e.g. H2S, SO2 and SO3, as well as nitrogen and ammonia. In an industrial atmosphere, a multi-component gas mixture is often found. In such cases, the formation complex, usually multilayer scale is observed. Chemical reactions that decide to the scale formation in these conditions are depending on many factor, which include:

* chemical affinity between the metal and the components of the corrosive atmosphere,
* the rate of formation of individual scale-forming compounds,
* properties of the compounds forming part of scale, above all their state of aggregation and solid solubility in solid state, and susceptibility to formation of multicomponent compounds,
* compactness or porosity of the formed scale and its permeability to the atmosphere and its components.

## Scale defect in the hot forming process

During the hot forming process, where the processed metal resides in the air, there is going to gas corrosion. The result of that process is the formation of a layer of scale on the metal surface. Because the scale layer separates the working rolls from the metal, its properties have a significant impact on the friction and heat exchange between the band and the work roll (Li and Sellars, 1996). Direct result of this interaction is impact on the condition of air and lifecycle of work rolls (Vergne and in., 2001). The parameters, which has impact on the formation of scales, that have been described in the literature are:

* temperature and time of rolling (Andorfer and in., 2003; Bolt and in., 2002; Chen and Yuen, 2000; Ginzburg, 1989; Sun, 2005; Sun and in., 2004),
* thickness of the rolled stand (Blazevic, 1996; Sun, 2005),
* rolling force (Andorfer and in., 2003; Bolt and in., 2002),
* chemical composition of steel (Tan and in., 2001)
* the silicon content in the steel alloy and the heating temperature in the furnace (Ginzburg, 1989)
* unequal cooling of the rolled strand (Chen and Yuen, 2000).

There are many types of defects related to the nature of the scale deposited on the surface of the rolled metal, such as: primary scale, secondary scale, rolled strip, red scale, etc.

There are several to remove scales in the hot forming process, including: mechanical cleaning, high pressure water jet cleaning, etching before cold rolling, flush annealing. In modern rolling mills often used method of removing (sapping) scales is high pressure water jet cleaning (Tiley and Munther, 2009). Pre-scaling is located behind the unloading zone of the rolling mill. After the ingot is released from the furnace, it is fed with a roller conveyor to the pelletizer to remove the primary scale formed during the heating in the furnace. The water pressure in the collector manifold depends on the type of installation and reaches 20 bar. Figure 7 is a pictorial drawing of two collectors of a scale ebullator. In the next steps of production, the tape is fed to the initial cage and the scale squeezer located in front of the group of finishing mills.

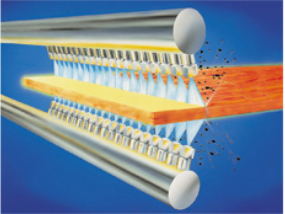


Figure 7: Collectors with nozzles to remove rolling scale

# Classification of defects and division of the scale

The goal of this chapter is to show division of scale defects based on vision character and reasons of occurrence. Divided scales was an input to classification system. To learn deep network, there was required to has large dataset with classified scale defects.

## Rolled-in primary scale

Rolled off scale *(Figure 8)*, also called sagger (Parsytec-AG, 2005) is a defect classified as surface defect, which surface is characterized by transverse, in relation to the direction of rolling and scale breaks. The size of defect varies from a few millimetres of wide to several centimetres of length. This class can be found locally, mainly at the beginning of strand, with different density of both on its surface. It often occurs with other type of scales, which has not been correctly removed. After the digestion process, the defect can occurs in the form of depression of surface.

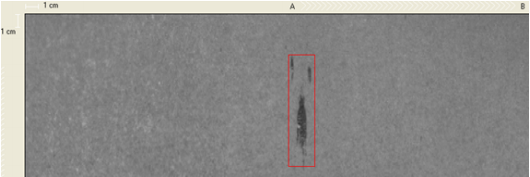


Figure 8: Rolled-in primary scale

This is a typical defect among the cases of rolled scale, which is created before the group of finishing frames. It is taken from insufficient removal of the original weight lying on the ingots after leaving the furnace. The reason of the formation of excessive scale on the ingots may be to high temperature in the furnace or a high concentration of oxygen in the furnace atmosphere. This leads to formation of layers of scale, which can be found to be incompletely removed by the scale squeezers. The appearance of the defect may be confused with a husk or the red scale. To avoid the formation of the scale, it is necessary to control compliance with the unloading temperature, the correctness of the operation parameters of the heating furnace of the ingot as well as the efficiency of descaling systems.

## Secondary Scale

The surface of a single defect *(Figure 9)* is characterized by a round, black shape with a diameter of 1-2 mm, and a long, white tail directed along the rolling line, which can reach length of up to 200 mm. Defects always appears in groups of different numbers, and white tails can overlap. The defects occur along the entire length of the band, on both sides, with grater severity at the beginning.

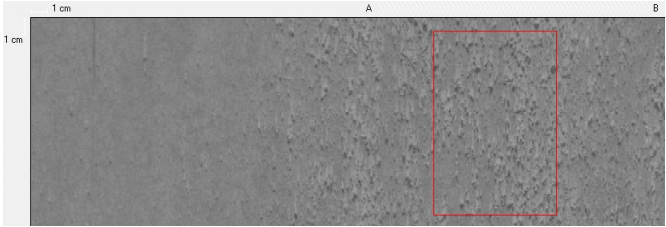


Figure 9: Secondary Scale

This type of defect comes from the rolling of scale by a group of finishing frames. Most often it occurs on thin sheets with a thickness below 2 mm at to high rolling temperature. Additional parameters, which causes the creation of defects are low-intensity cooling of rolls and strand as wear of rolls. When the look of the defect is not prolonged, it can be assumed that it was created on one of the middle finishing frames. Defect detection, and classification can be difficult due to defects giving different region of interest (ROI). The defect is erroneously classified as defect which looks like letter *V*. To prevent formation of defect, it is necessary to control compliance with the rolling temperature in the production of thin sheets (below 2 mm) and the parameters of the group of finishing mills, in particular the cooling performance of the rolls. An increase in the occurrence of a defect may indicate a less efficient operation of the descanters of the initial mill and the group of finishing mills.

## Rolled *“V”* scale

Rolled scale which looks like the letter *V* is a kind of defect, which originate from the group of finishing frames *(Figure 10)*. Its shape resembles the letter V, which width and height reaches a maximum of 20 mm. Typically, the defect appears in numerous groups, on the upper surface and the beginning of the band, but may appear on its entire length.

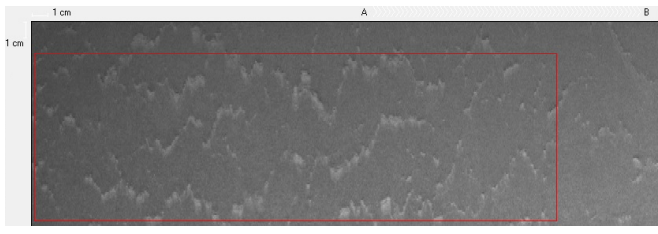


Figure 10: V scale

Like the rolled scales, this type of defect comes from the group of finishing frames. Most often it occurs on thin sheets below 2 mm with too high rolling temperature. As in the case of rolled-up secondary scale, additional parameters affecting defects are low-intensity cooling of rolls and strand, and wear of rollers. When the look of defects is not extended, it can be assumed that it was made in one of the middle finishing frames. Additionally, the origin can be related with the chemical composition of steel. Steels containing phosphorus are particularly susceptible to this defect. In order to prevent forming of a defect, it is necessary to control compliance with the rolling temperature in the production of thin sheets, as well as the parameters of the group of finishing mills. Increasing the occurrence of a defect may indicate a less efficient operation of scale desiccation rollers and a group of finishing mills.

## Peeled roll scale

The cause of this defect *(Figure 11)* is a slight loss of working roll caused by thermal or mechanical damage of its surface. The other name of this defect is a *black skin scale*. It has an oval shape and consist of small, black points with a diameter a round of 1-2 mm. Defect can occur on both sides of band, along the whole length in characteristics stripes.

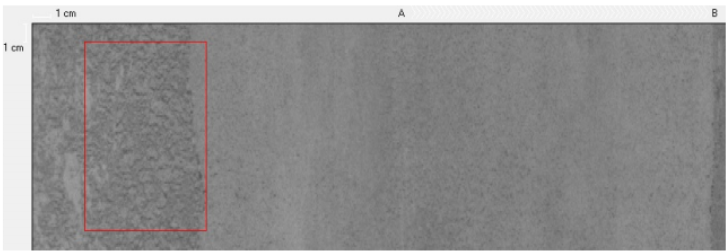


Figure 11: Peeled roll scale

The disadvantage usually arises on the first frames of the finishing group. The immediate cause is the wear of the rollers, for example during the rolling long sheet metal of a short length. The appearance of this defect means that the working rolls will have to be sanded. The defect is discovered in large area of ROI. Its classification is sometimes confused with Secondary Scale. In order to avoid forming the defect, it is necessary to control compliance with the rolling temperature in the production of thin sheets, with particular emphasis on maintaining the lowest possible ending temperature of the bands from the group of finishing mills. It is also necessary to monitor the parameters of the group of finishing mills, and in particular to meet the required reduction in band thickness. Increasing the occurrence of a defect may indicate about deterioration of the surface condition of work rolls, or less efficient work of mill scale descaler, and groups of finishing mills.

## Red scale / Tiger scale

According to work (Fukagawa i in., 1994) red scale *(Figure 12)* is a type of scale formed due to the high content of silicon in the steel alloy. It is precipitated in the furnace before the rolling process. After rolling, it turns red. The defect consists of long and irregular stripes and has a noticeably greater harshness than the rest of the surface of the strand. Its length reaches several centimetres. It can occur on whole length of band, on the both sides.

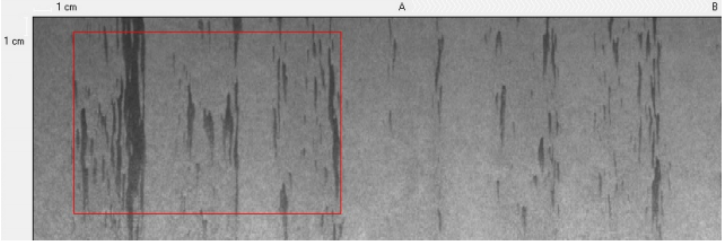


Figure 12: Red Scale

Defect appears on steels with a silicon addition in the chemical composition. Usually the defect will be removed during the etching process (Breitschuh i in., 2007). In order to monitor the occurrence of defects, verification of the rolled steel grade should be performed. If it is silicon steel, then check the level and range the presence of scale. If it is not, the chemical composition of steel should be performed in order to verify the correctness of its casting. Defect detection tends to broadcast every large ROI, often covering the entire length and width of a single image. Defect classification can be confused with husks and rolled-in primary scale.

## Heavy scale

Heavy scale *(Figure 13)* is intensified variant of rolled-in primary scale, which formation is caused by the improper operation of scale squeezers. The entire ROI area is covered a clear and thick scale. The defect size is between from 500 mm to 2000 mm (on band width) or 20000 mm (on the band length). Its can occur on both sides of band, and it entire length. Numerous clusters of defects can be observed at the beginning of band, or in places where the scaling of the scale started to work incorrectly.

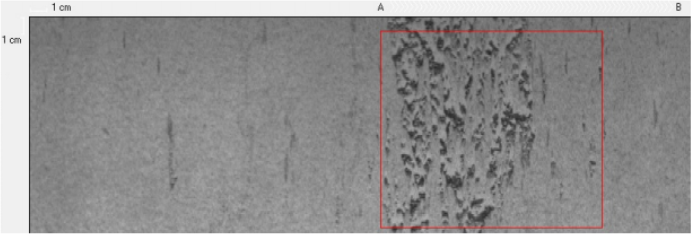


Figure 13: Heavy Scale

Defect may appear on all types of steel in the course of improper operation of the scale squeezer. The classification may not be correct, because of the possibility of confusing the defect with the rolled-in primary scale. In order to avoid formation of defect, first of all, the efficiency of scale descaling systems should be controlled. Additional parameters, that may affect the level of defects, there are parameters of heating the ingot in the furnace and the inflated temperature of its unloading.

## Bad descaling

Bad descaling, called also Finishing Scale Breaker (FSB), arises due to damage or abnormal operation of scale squeezers located in front of group of finishing mills *(Figure 14)*. It can be composed from the different scales. The nature of this defect clearly indicates to need to stop the rolling process and check the descaling system. The defect has the shape of longitudinal strips of scale, which appear in the places of improper operation of the knockout nozzles. The length of strips is from a few centimetres to even several hundred meters. The width of the defect depends on the number of the number of non-operating nozzles of scale squeezer. Defect can appear on whole length and width of band, on both sides.

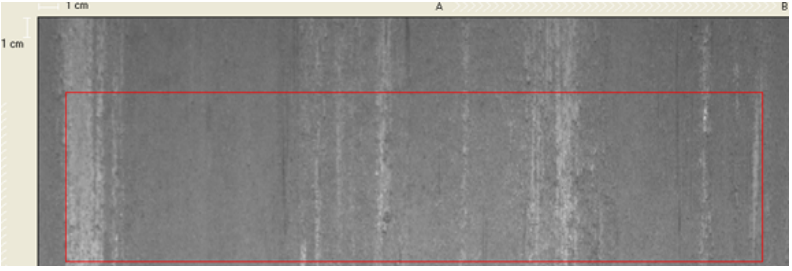


Figure 14: Bad descaling

In order to avoid creating a defect, all parameters related to the correct operation of descaling systems located in front of a group of finishing mills should be controlled. In particular, it can be talk about the correctness of the operation of the sensors and automation of pounders, pressure and flow of water on the nozzles of beaters, patency of the nozzle nozzles, and the speed of the rolled stand in relation to the performance of skiners.

According to describe above, bad descaling is defect, which consist of other scales, which have not been properly removed by the pounders. Moreover, the information about the occurrence of this defect does not come from one image, and it could not classified by different decision systems. Therefore, the defect is not a subject to direct classification by AI methods.

# Methods of recognition and classification of images

Image recognition is in other words the extraction of useful information from it, assign the spaces of features and assign it to a class. Construction of classification model using AI methods is also called machine learning, which can be divided into supervised learning and unsupervised learning. Supervised learning requires some knowledge about the set problem and initial designation of classes to which objects will be assigned. Unsupervised learning requires the use of clustering methods, relying on grouping similar objects inside the same cluster, thus determining the relevant from the point of view of the class problem posed.

Next division is the division due to the knowledge of the distribution of probabilities of belonging to the points of the space of features to classes. In the absence of that knowledge, it is a non-parametric classification, in otherwise it is a parametric classification.

Another division related to making decisions on the affiliation of unknown objects to the distinguished classes is the kind of approach to making decisions. It can be holistic or structural. Holistic approach takes into account all vision features (space of features) of the recognized image and decides about its membership in one decision act. The structural approach uses features to determine the elements and mutual relations of the posed problem, build its structural description, and finally makes the classification based on the built-in description. It should be mentioned about the syntactic methods referring to the concept of primitivism, which is the basic structural element of the object or scene being studied. In syntactic methods, recognition occurs as result of the parsing of properly constructed deterministic grammar, in which basic lexical units correspond to the identified primitives. Depending to the complexity of grammar, string, graph and tree methods are distinguished. The basic difficulty in the recognition process is the generation of an appropriate grammar.

## Minimal distance methods

Minimal distance methods are based on the geometry of feature space. They strive to link the membership function *C* with distances in a defined space *X* equipped with the appropriate metric *ρ*. In general terms, it will be a mapping:

.

Next, the distances of the recognized object are sought from other objects of known affiliation, that is from the example images of the training sequence. Tested object will be assigned to this class as the closed element of the training sequence (in the sense of the adopted and defined metrics). The proper metric are chosen of empirically by means of trial and error or by making a completely arbitrary decision.

To popular minimal distance methods can be included various varieties of the method of nearest neighbour (*NN*). It assumes that, for defined some metric *ρ* in a space *X* the following mapping record is possible *C:*

where element xi,kbelongs to a subset of the training sequence Ui. A positive constant ε was introduced in order to eliminate the possibility of obtaining an infinitive value of the function Ci(x). NN method assumes selection of the xi,k element according to the rule:

.

In practice, this means the classification of an unknown object (marked with a green star in the image) to the closest image in the space of training sequence *(Figure 15)*.

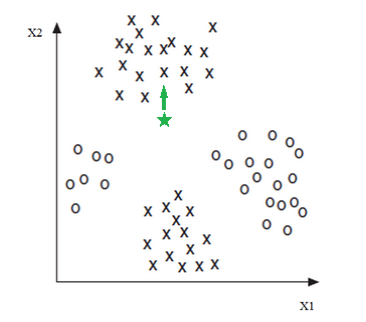


Figure 15: The rule of making decisions in the NN algorithm. Crosses and circles mean grouped elements of the space of features that indicate the belonging of groups of features to a given class

There are different variations of the NN method. It is worth to mentioning here the *k*NN method, which classifies an unknown object based on the predominance of the number of objects with a known classification in the neighbourhood of the k-nearest neighbours. Figure 16 shows the idea of the *k*NN algorithm. Each new object (green star) can be classified to the one of the two classes (crosses and circles). If *k=3* (yellow circle) then new object will be assigned to the circle class, because there are more than 3 objects in its neighbourhood than the crosses. If *k=15* then object will be assigned to the crosses class, because there are 8 crosses and only 7 circles inside the red line circle.

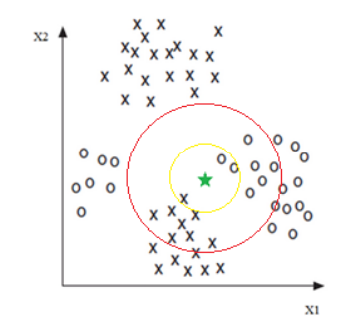


Figure 16: Decision-making rule for the kNN algorithm.

## Approximation methods

Approximation methods determine the function of belonging to *C* on the basis of developing it in a series relative to fixed family of *φ* function described by dependence:

,

where used base functions create an ordered family *ϕ*, such that:

.

Development coefficients Vvi (i=1, …, L; v=1, 2, …), called are weights and determinate a specific function Ci(x). Determining the family of functions *ϕ* is the most difficult application of this method in practice. Correct determination of this family *ϕ* allows replacing the task of looking for decision rules with the task of determining the value of weights. Finally, it guarantees fast recognition with relatively small memory and computing power requirements.

## Probabilistic methods

Probabilistic methods allows a different approach to determining the function of belonging. Here the traits are treated as statistical observations, and the recognition process is similar to the verification of statistical hypotheses.

Based on work (Tadeusiewicz i Flasiński, 1991) probabilistic methods assume, that the data are a priori probability of occurrence of belonging objects to individual class: p1, p2, … , pL, where:

,

and conditional probability density distributions P(**x**/1), P(**x**/2), …, P(**x**/L) being the probabilities of the vector x occurring on the assumption that the objects belongs to class i. In addition, the concept of generalized loss Q(**x**) is introduced, which is expected in the case of the appearance of an object described by a set of features x, which value Q depends on the chosen membership function C. Usually, this relationship has a minimum, the achievement of which can be equated with the choice of the optimal recognition technique. It can be further built the quality assessment Q(A), recognition algorithm Â, and then look for this form of membership function C, to minimize Q(A).

A particular and widely discussed in the literature (Bishop, 2006; Duda i in., 1996; Webb, 2002) case of probabilistic methods for determining the membership function *C* is Naive Bayes classifier. It assumes the independence of the futures of the objects with respect to each other, which means that hat each N-dimensional density of the probability distribution can be presented as product of density of probability distributions for single variables. This simplification frees from the knowledge about multidimensional decomposition densities, and problem is simplified to the density of one-dimensional distributions. The downside of this simplification is the deterioration of the overall classification in case the actual system has variables are not independent of each other.

The methods of determining the membership function should be supplemented with the special methods, which include the potential methods, stochastic approximation, and described in later chapters: decision trees, supporting vector methods, and recognition using neural networks.

## Potential methods

Potential methods build membership function *C* as superposition of function *K(****x,xi****,k)*, that in their shape resemble the distribution of the electric potential around the point charge. These functions are strongly diminishing between the point **x**i,k, generate potential and belonging to the learning sequence Ui, and point **x**, for which the value of the potential function is calculated. In literature, these function can be founded:

,

or

,

where *µ* is a positive constant, and *ρ* is an any metric. After selecting the potential function, the membership function can be saved as:

,

where *ηk* is often omitted, a numerical sequence.

## Stochastic approximation

Described in the work (Robbins and Monro, 1951), the stochastic approximation determines the zero points (roots) of the regression equation in the form:

,

where **ϕ** is a function of vector argument **x**, which is assumed to be the implementation of a stochastic process, and **E** is the expected value. Practical use the method to the image recognition requires the identification of function **ϕ**, with the some concept in the field of image recognition. It can be done by entering a separating function, which indicates the correct class i I through its sign. Taking as an example recognition for only two classes with numbers i1 I and i2 I, and assumes that each object d D must belong to one of the classes, membership function C can be designated as:

.

Function determinates the membership of object based on the character:

## Decision trees, DT

Decision trees, called DT in short, are graphical method of decision support, and one of the better-known and frequently used SI methods. Decision trees are based on the idea of dividing a complex problem into a simpler one, and then using the same split rule for newly created subtasks. In theory describing decision trees, the following concepts are distinguished:

* nodes – store tests that check the values of the features of the examples,
* branches – connect successive nodes and leaves, correspond to the results obtained during the testing tests,
* leaves – store categories, they are an end element, so no branches come out of them,
* root – the “lowest” level of tree, being a node or leaf, that has no parent nodes.

Decision trees are defined as directed acyclic graphs in which each top is decisive top or leaf. Leaf represents the object belonging to the class. Classification consists in asking questions, going each time for response, along successive nodes from root to leaf. The leaf on which the questioning will end is about assigning the class label to the object. There are different algorithms developed on the basis of decision trees, and the more popular ones are CART (Breiman i in., 1984), „Interactive Dichotomizer” (Quinlan, 1979) and C4.5 (Quinlan, 1993).

A very simple sample tree classifying the type of sport that is recommended for spinal or knee ailments is shown in Figure 17.

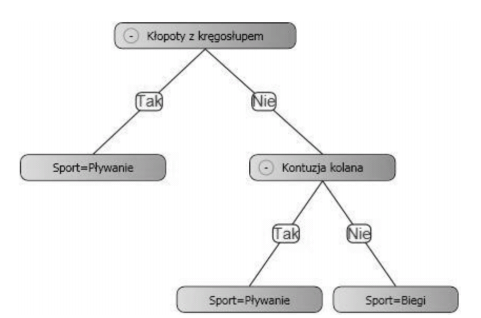


Figure 17: Example of a simple decision tree

The formal definition of decision tree, according to work (Cichosz, 2000), assumes that the field is *X*, on which the features *a1*, *a2*, …, *an* and a class of concepts *C* about the category collection K. It can be said that every leaf containing any category label d K is a decision tree. Then it assumes that, if is a test carried out on the values of the features of examples with a set of possible results and T1, Tn, …, Tm are decision trees, is a node containing test *t* from which branch comes out, for i=1,2,…, m  
the i-th branch corresponds to the ri and leads to the Ti tree, it is a decision tree.

Problems related to the creation decision trees are of different nature. One of them is an apex fission criterion, which tells us about construction of such number of vertices, so that each of them contains elements as diverse as possible. In practice, this means the construction of inefficient trees.

Another problem is ability to overfitting of decision tree model, that is, the choice of such classification model whose very good quality of classification on the training set does not translate ability to generalize and predict labels from the test set. In that case the cutting a tree is used.

Problems associated with decision trees can also include instability of the algorithm, characterized by high variability of the tree structure caused by small changes in input parameters, and a tendency to duplicate the sequence of questions in separate subtrees (so-called replication).

An interesting variation of classification trees are reinforced classification trees. This method consists in creating a sequence of simple trees, each of which is used to predict the rest of the previous tree. The computational approach of reinforced classification trees is also known under the names: TreeNet ™ (Salford Systems, Inc.), MART (Multiple Additive Regression Trees), GBM (Gradient Boosting Machine), GBDT (Gradient Boosted Decision Trees), GBRT (Gradient Boosted Regression Trees).

## Artificial neural network, ANN

Artificial neural network, called briefly ANN, is a generic name for a simulation program or an electronic circuit implementing AI tasks, through a set of elements called artificial neurons. These elements model in a simplified way the operation of real neurons, that means elements of brain.

Each neuron has many inputs that are equivalent to synapses in the brain, and one output that simulate the axon. Numeric coefficients wi (i=1,2,…,n), called synaptic weights characterize each of the inputs of each cell in the network. The concept of threshold ϴ and stimulation e is introduced, which is described by the formula:

,

Figure 18 presents the model of the unit performing the above calculation. Function:

,

is called the activation (or transition) function. The behavior of the neuron is strongly dependent on the used activation function. Linear, sigmoidal and unit-based functions are often used.

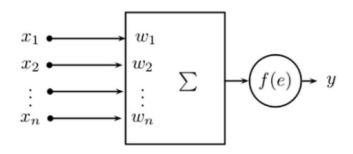


Figure 18: The model of the unit performing the neural network calculations

Neuron network is a set of neurons connected by layers. Each network has inputs connected with external observations and outputs that show the results of calculations. There Between the input and output layers, there may be n-layers of hidden elements, which cannot be directly observed from the entry or exit side. There are two network structures: feedforward and feedback. Feedforward networks have one direction of signal flow. Feedback networks have feedback *(Figure 19)*.

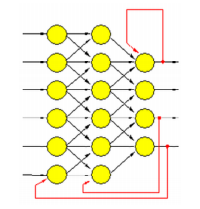


Figure 19: Structure of an example of a neural network. The red line indicates the extension of the normal neural network to the feedback network.

Neural network training process is a resultant of individual actions neurons, and interactions between them. Back error propagation algorithm is a frequently used training algorithm, which systematically reduces the error made during the training process. The result is a gradual improvement of the network. There should be mentioned about the phenomenon of overfitting, consisting in excessive adaptation of the network to the training set (low error) which can cause large errors on the test set. Parameters affecting the final form of the neural network are:

* training cycles - defines the number of cycles used to the training of network
* learning rate - speaks of the level of weight changes in each training cycle
* momentum - refers to the "inertia of the learning process". When the momentum parameter is used, the correction of neuron weights depends not only on the input signal and the error made by the neuron, but also from the correction of weights in the previous learning step. It affects the slowing down of the training process at the final stage of the training process, at the same time increasing its fluidity.

The advantage of neural networks is the lack of the need to have knowledge about the relationship describing the given regularity. Therefore, neural models can be used wherever the exact law describing the development of the studied relationship is unknown. Additionally, models of neural networks has adaptive character. It can be used to describing of relationships that change over time. When the new data appears, the network can be doubled, which includes the inclusion of new information in the current model.

The disadvantages of neural network include the problems related to the selection of the proper network structure, which is often done on the principle of trial and error. This results in a significant increase in the construction time of the model. Additionally, neural networks often cut off the creator from the possibility of direct interpretation of model coefficients.

## Support vector machine, SVM

The purpose of the classification using the SVM methods is to determine in the space the features of the optimal separating hyperplanar, with the maximum margin.

In the elementary approach of the SVM model, the existence of a training set {xi,yi}, xi ϵ Rd, yi ϵ {-1,1} is assumed, which elements belongs to one of the classes marked as {-1,1}. The goal of that model is to determine the decision limit (hype-plane) between these classes. Equations of hyper-planes can be written for linearly separable classes, whose normal w and distance from the centre of the system b meet the conditions (Schӧlkopf i Smola, 2000):

.

Graphic interpretation of the above formulas is shown in Figure 20.

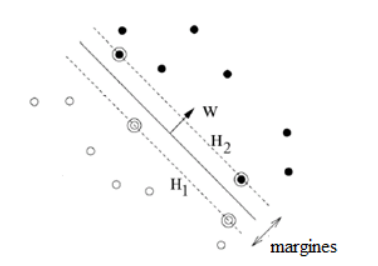


Figure 20: Hyperplane that divides two linearly separable classes in the SVM model

Next, the margin m between these hyper-planes can be determined, which in accordance with the SVM assumptions is subject to maximization. Formally, the margin is noted as:

.

The hyperplane equation can be replaced with a single equation:

.

The concept of regulatory constant C is introduced in the SVM classifier. If the value of constant C is higher, then the matching of the model increases, on the other hand, its generalization properties are falling and conversely. To obtain a decision function, the margin is maximized by the minimization of .

In the case of non-linear separability, the fact that the learning algorithm depends on the product of vectors is used and it is possible to use the kernel function in the place of the vector product. It gives the possibility of mapping the spaces of features into a space with a larger number of dimensions, which in turn allows you to change a non-linear problem in a given space of features into a linear problem in a space with a higher dimension (*Figure 21*). This information is more widely known as the kernel trick (Bishop, 2006).

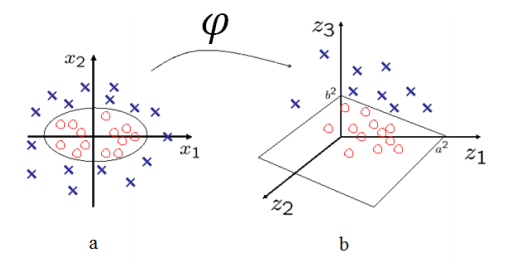


Figure 21: Kernel trick: a) Decision limit in the original space of features; b) Decision-making after transformation

The formal form of the decision function of the SVM model after transformation using the kernel function takes the form of:

,

where is a kernel function. Many kernel functions can be distinguished, and the more popular ones are:

* gaussian

,

* polynomial

,

* sigmoidal

,

Depending on the adopted kernel function parameter γ, which can be identified with the "width" of the kernel function, will be subject to parameterization.

The advantage of SVM is possibility to application of multidimensional transformations (*kernel trick*), which using the proper kernel function translates into high efficiency of the model in practical applications. In addition, the application of the margin maximization rule gives the opportunity to more easily generalize the problem. It can also be said that proper parameterization of SVM allows to eliminate the influence of outliers of the training sequence and increase capabilities generalization of model.

SVM's minuses include slow training, particularly annoying with a large amount of data in the training sequence (Burges, 1998). The SVM solutions are often complicated. One of the basic limitations of SVM is the problem of proper selection of the kernel function.

## Deep Neural Networks

Image recognition problem can be divided into two steps: define the set of classes, and train model which can recognize images using photos from delivered dataset. Unfortunately there are multiple factors, which can affect to the stable of model. If image is treated as raw of pixel data, problem can occur as: position of recognized object, various background on the picture with object, position of camera,  
image sharpness (Figure 22) or similar objects (*Figure 23*). All these factors affect to the RGB values of pixels. In basic approach each pixel can be connected to one neuron. However, to prepare good model, its requires a lot of images, which are an input to neural network. Thousands or millions of neurons are computationally expensive, therefore classic artificial neural network are not proper approach of image recognition.

|  |  |
| --- | --- |
| 5_1  Figure 22: Many variations of recognized objects | 5_2  Figure 23: Similar objects |

To handle problem of image recognition using artificial neural network, Convolutional Neural Network will comes with help. Convolutional Neural Network known as CNN or ConvNet is a deep, feed-forward artificial neural network. Feed-forward neural networks, also called multi-layer perceptrons (MLPs) are the essence of deep learning models. Information between the layers flows through the model, without any feedback connections.

Usage of convolutional neural network allows to progressively extract higher- and higher-level representations of the images. Raw pixels of image is taken as input for the CNN, and there are used to learn how to extract feature like (e.g. shapes), and classify the object. There is no needed preprocessing of data, which can derive features like shapes and textures.

To start, input feature map is needed, which contains three dimensional matrix, where first two dimensions describe the length and width of image, and last correspond to channel of image color (size of last dimension is equal to 3). Next, CNN performs three operations: Convolution, Activation Function (most common function: ReLu) and Pooling.

*Convolution* stage divides the input to feature map, applies the filters. As a result is output of feature map, which size and depth can be different then in input data. Filters in convolution moves over the input of grid horizontally and vertically, one pixel at time, extracting each element of output feature map. There are performed multiplication for each pair of filters and pair of elements, which are sum to get a single value. It is executed to create the convolved feature matrix

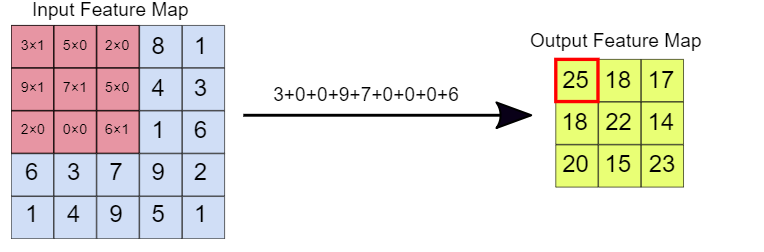


Figure 24: Convolution of input data to obtain convolved feature matrix

To extract the features (edges, shapes, textures), CNN learns optimal values for the filter matrix during the training process. To increase the numbers of features which CNN can extract, the number of filters should be increased. For the other hand, the time of training is increased too, because the filters compose the majority of resources expended by the CNN.

After each convolution operation, the CNN performs *Activation Function* stage, which decides that the input is putted into the node. The most popular method is *Rectified Linear Unit (ReLu)*. ReLu is used to introduce nonlinearity into the model, and it can be described by following equal:

The next step of training process is pooling, which is a form of non-linear down-sampling. It is performed to reduce the number of dimensions, at the same time does not lose the most critical feature informations. Usually the *Max Polling (Figure 25)* algorithm is used.

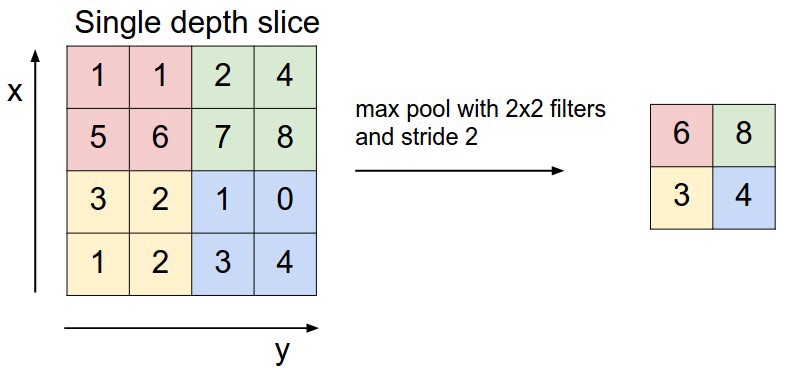


Figure 25: Max polling

Max pooling algorithm is similar to convolution stage. Algorithm are moving on the feature map, and creates the new map, with specified size. From the input map, the maximum value is taken and putted as output of new feature map. Other values are discarded.

The last stage is called *Fully Connected Layers*. It is placed at the end of CNN and it can be one or more layers (for the two *fully connected* layers, each node from the first layer is connected to each node in second layer). The goal of this stage, is to classify object depend on features obtained during the convolution.

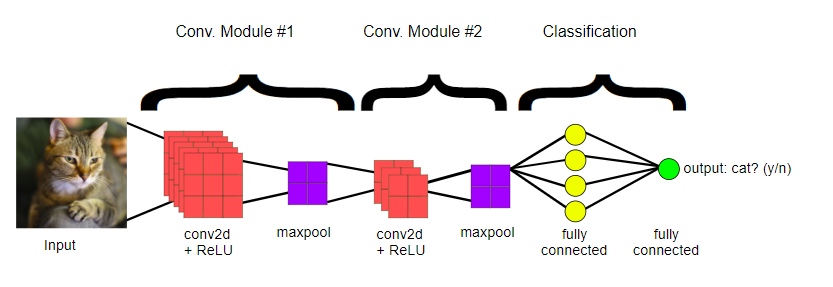


Figure 26: Convolutional neural network, divided on described steps

Usually, fully connected layers contain *Softmax* activation functions. This function calculates the probabilities distribution of the event over ’n’ different events. According to image classification, function calculate the probabilities of belonging object to each class, which is helpful to determining the target class for the inputs.

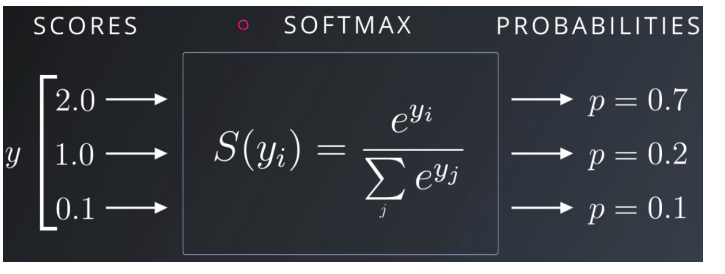


Figure 27: Softmax activation function

# Image classification system based on CNN

The aim of thesis is to create system, which can detect and classify defects created during hot forming process. To achieve the goal, Image classification system based on CNN was created. Convolutional Neural Networks was chosen, due to good dealing with the classification problem. Images and information about them was used as input, which is being pre-processed. Next pre-processed data are used as input to Network Model, which trains and try to learn, how to classify defects. At the end of the training, model with learned weights is saved, and the accuracy and loss metric, from each epoch also are saved. The metrics are delivered to Result functionality of System, which prepare data as graph. Graph is a form which are readable, and it is a good form to evaluate the results of training. Figure 28 shows the architecture of described system.

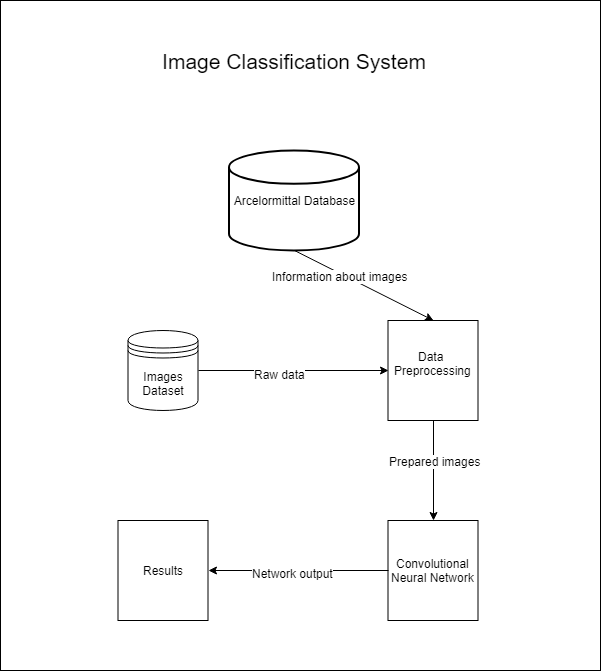


Figure 28: Architecture of Image Classification System

## Data pre-processing

To classify defects using CNN, input data has to be in proper form. Images should be aggregated by class of defect, and each image has to have the same size and color channel. Data delivered from arcelormittal was not aggregated by class of defect, but it was aggregated by coil identity, camera identity, and defect number. Path of each file can be described by pattern: ***coilId/cameraId/*** ***P6810010\_coilId\_cameraId\_srcimg\_defectNo.*** Based on that pattern, images could be assigned to corresponded record from database. Records are stored in *defects* table, which contains all important fields to build CNN dataset. To prepare data pre-processing step, scripts in python 2.7 were written and following external libraries were used: *Sqlalchemy, OpenCV 2*. *Sqlalchemy* was used to establish connection with database, and *OpenCV* 2 was used to transform images.

First step in data pre-processing is getting all defect classes and all defects from database. To establish connection and execute queries, system uses *DatabaseHandler.py (Code Block 1)*. *DatabaseHandler* is a separate python class, which can connect to many types of SQL databases. It uses Sqlalchemy, which is a common python Object-relational mapping (ORM) library.

|  |
| --- |
| **from** sqlalchemy **import** \* **import** ConfigParser **import** tables **from** sqlalchemy **import** func  SECTION = **'Database'  class** DatabaseHandler:   **def** \_\_init\_\_(self):  config = ConfigParser.ConfigParser()  config.read(**"./res/config.ini"**)  self.USERNAME = config.get(SECTION, **'Username'**)  self.PASSWORD = config.get(SECTION, **'Password'**)  self.URL = config.get(SECTION, **'Url'**)  self.PORT = config.get(SECTION, **'Port'**)  self.DB\_NAME = config.get(SECTION, **'DbName'**)  self.DB\_TYPE = config.get(SECTION, **'DbType'**)  self.engine = create\_engine(self.getDbSQLPath())  self.engine.connect()  self.defectsTable = tables.t\_defects  self.defectsClassesTable = tables.t\_defectclasses  **pass   def** getDbSQLPath(self):  **return** self.DB\_TYPE + **'://'** \  + self.USERNAME + **':'** \  + self.PASSWORD + **'@'** \  + self.URL + **':'** \  + self.PORT + **'/'** \  + self.DB\_NAME   **def** getDefectRecordsToGetFile(self):  query = select([  self.defectsTable.c.coilid,  self.defectsTable.c.camerano,  self.defectsTable.c.defectno,  text(**'defects.class'**)  ])  **return** self.engine.execute(query)   **def** getROIOfRecord(self, coilId, cameraNo, defectNo):  query = select([  self.defectsTable.c.roix0,  self.defectsTable.c.roix1,  self.defectsTable.c.roiy0,  self.defectsTable.c.roiy1  ]).where(and\_(  self.defectsTable.c.coilid == coilId,  self.defectsTable.c.camerano == cameraNo,  self.defectsTable.c.defectno == defectNo  ))  **return** self.engine.execute(query).fetchone()   **def** getDefectClasses(self):  query = select([  self.defectsClassesTable.c.classid,  self.defectsClassesTable.c.name,  ])  **return** self.engine.execute(query)   **def** getDefectString(self, defectClass):  **return** (str(defectClass[1])).replace(**' '**, **'\_'**)   **def** getDefectClassesString(self):  defectClassesStrings = []  **for** defectClass **in** self.getDefectClasses():  defectClassString = self.getDefectString(defectClass)  defectClassesStrings.append([defectClassString, defectClass])  **return** defectClassesStrings   **def** getDefectClassDefectStringDictionary(self):  defectsDict = {}  **for** defectClass **in** self.getDefectClasses():  defectsDict[defectClass[0]] = self.getDefectString(defectClass)  **return** defectsDict |

Code Block 1: DatabaseHandler.py

System get all class of defects, and put them to dictionary, where the key is an id of defect, and the value is a name (*Code Block 1*, method: *getDefectClassDefectStringDictionary)*. Next system get all defects (*Code Block 1*, method: *getDefectRecordsToGetFile*), and iterate through them. During iteration system try to match proper record with image, from dataset (*Code Block 2*, method: *getImageBasedOnDatabaseRecord*). Matching is performed by *ImagesHandler* class (Code Block 2).

|  |
| --- |
| **from** imutils **import** paths **from** keras.preprocessing.image **import** img\_to\_array **import** cv2 **import** os  FILE\_NAME\_PREFIX = **'P6810010'** IMAGES\_DIR = **'./res/photos'** DIR\_DELIMITER = **'/'** FILE\_NAME\_DELIMITER = **'\_'** FILE\_EXETENSION = **'.jpg'  class** ImagesHandler:   **def** \_\_init\_\_(self):  **pass   def** showImage(self, img):  cv2.imshow(**'image'**, img)  cv2.waitKey(0)  cv2.destroyAllWindows()   **def** saveImageToFile(self, img, dirpath, filename):  filepath = dirpath + **'/'** + filename + FILE\_EXETENSION  cv2.imwrite(filepath, img)   **def** getImageBasedOnDatabaseRecord(self, coilId, cameraId, defectNo):  **return** self.openImage(self.getFilePath(coilId, cameraId, defectNo))   **def** openImage(self, path):  **return** cv2.imread(path)   **def** getFilePath(self, coilId, cameraId, defectNo):  **return** IMAGES\_DIR \  + DIR\_DELIMITER \  + str(coilId).zfill(8) \  + DIR\_DELIMITER \  + str(cameraId).zfill(2) \  + DIR\_DELIMITER \  + FILE\_NAME\_PREFIX \  + FILE\_NAME\_DELIMITER \  + str(coilId).zfill(8) \  + FILE\_NAME\_DELIMITER \  + str(cameraId).zfill(2) \  + FILE\_NAME\_DELIMITER \  + **'srcimg'** \  + FILE\_NAME\_DELIMITER \  + str(defectNo).zfill(4) \  + FILE\_EXETENSION   **def** mapDirToNumber(self, path):  number = 0  numbderDirMap = {}  **for** dir **in** sorted(os.listdir(path)):  numbderDirMap[dir] = number  number += 1  **return** numbderDirMap   **def** getImgesAndLabelsInPath(self, path):  images = []  labels = []  classes = set()  imagePaths = sorted(list(paths.list\_images(path)))  labelsMap = self.mapDirToNumber(path)  **for** imgPath **in** imagePaths:  image = cv2.imread(imgPath)  image = img\_to\_array(image)  images.append(image)   label = imgPath.split(os.path.sep)[-2]  labels.append(labelsMap[label])  classes.add(label)   **return** images, labels, len(classes) |

Code Block 2: ImagesHandler.py

Additionally system checks multiple conditions, which are related with image size and will be used to create a lot of different datasets. At the end method returns images, related class to image, number of records and counter of each defect class. All operations are shown in *Code Block 3*.

|  |
| --- |
| **def** getMatchedDefects(no\_up\_rescale=False, onyl\_square=False):  matchedImgs = {}  matchedRecords = {}  defectsCounterDict = {}  counter = 0  skipped = 0  **for** defect **in** dbHandler.getDefectRecordsToGetFile():  img = imgHandler.getImageBasedOnDatabaseRecord(defect[COIL\_ID], defect[CAMERA\_ID], defect[DEFECT\_NO])  **if** img **is not** None:  **if not** no\_up\_rescale **and not** onyl\_square:  size = dbHandler.getROIOfRecord(defect[COIL\_ID], defect[CAMERA\_ID], defect[DEFECT\_NO])  x = size[X1] - size[X0]  y = size[Y1] - size[Y0]  **if** no\_up\_rescale:  **if** x < 32 **and** y < 32:  skipped += 1  **continue  if** onyl\_square:  **if** x \* 2 < y **or** y \* 2 < x:  skipped += 1  **continue** matchedImgs[counter] = img  matchedRecords[counter] = defect  counter += 1  **if** defect[DEFECT\_CLASS] **in** defectsCounterDict:  defectsCounterDict[defect[DEFECT\_CLASS]] = defectsCounterDict.get(defect[DEFECT\_CLASS]) + 1  **else**:  defectsCounterDict[defect[DEFECT\_CLASS]] = 1  **if** counter % 1000 == 0:  **print 'Got '** + str(counter) + **' records and skipped '** + str(skipped)  **return** matchedRecords, matchedImgs, counter, defectsCounterDict |

Code Block 3: Match images with database records

After matching process, system prepares directories in proper path, where each directory is a class of defect. Pre-processed data are saved in that directories, and CNN module uses data from that location. Sometimes number of classes were limited, and it was related with preparation of many variations of different datasets. Both operations are shown in   
*Code Block 4.*

|  |
| --- |
| **def** createClassFolders(defectClasses, dir, processedClasses):  **for** key, value **in** defectClasses.iteritems():  **if** key **not in** processedClasses:  **continue** path = dir + **'/'** + value  **if not** os.path.exists(path):  **print 'Creating '** + path  os.makedirs(path)  **return  def** getClassesSet(classesCounterDict):  classes = set()  i = 0  **for** key, value **in** sorted(classesCounterDict.iteritems(), key=**lambda** (k, v): (v, k), reverse=True):  **print** key, value  **if** i < CLASS\_NUMBER:  classes.add(key)  i += 1  **else**:  **return** classes |

Code Block 4: Create Class directiories, and limit number of defect classes

Next step is data transformation. In this step, defects are extracted from whole image. Information about place of defect are saved in database, and it is called Region of Interest (ROI) parameter. ROI is extracted from image, then new image is normalized to a common size. This operations are realized by *ImagesTransformator.py* class (*Code Block 5*).

|  |
| --- |
| **import** ConfigParser **import** copy **import** cv2  SECTION = **'Image'  class** ImagesTransformator:   **def** \_\_init\_\_(self):  config = ConfigParser.ConfigParser()  config.read(**'./res/config.ini'**)  self.WIDTH = int(config.get(SECTION, **'Width'**))  self.HEIGHT = int(config.get(SECTION, **'Height'**))  **pass   def** createImageBasedOnROI(self, image, x0, x1, y0, y1):  img = copy.copy(image)  **return** img[y0:y1, x0:x1]   **def** standarizeImage(self, image):  **return** self.resize(copy.copy(image), (self.WIDTH, self.HEIGHT))   **def** resize(self, image, size):  **return** cv2.resize(image, size) |

Code Block 5: ImagesTransformator.py

Extracting of ROI is an optional operation, which was changed according to creating new datasets. After this operations, transformed image is assigned to their class of defect. At the end transformed images with information about class are returned. This functionality is shown in Code Block 6.

|  |
| --- |
| **def** preprocessData(defects, defectsImgs, defectsCounter, processedDefects, normalize=False, resize=False):  i = 0  skipped = 0  processedImages = []  **print 'Start processing images...'  while** i < defectsCounter:  defect = defects[i]  **if** skipped % 100 == 0:  **print 'Skipped '** + str(skipped) + **' images'  if** defect[DEFECT\_CLASS] **not in** processedDefects:  skipped += 1  i += 1  **continue** defectImg = defectsImgs[i]  **if** normalize:  roi = dbHandler.getROIOfRecord(defect[COIL\_ID], defect[CAMERA\_ID], defect[DEFECT\_NO])  defectImg = imgTransformator.createImageBasedOnROI(defectsImgs[i], roi[X0], roi[X1], roi[Y0], roi[Y1])  defectImg = imgTransformator.standarizeImage(defectImg)  **if** resize:  defectImg = imgTransformator.resize(defectImg, SIZE)  i += 1  processedImages.append([defectImg, defect[DEFECT\_CLASS]])  **if** i % 100 == 0:  **print 'Process '** + str(i) + **' of '** + str(defectsCounter) + **' total images'  return** processedImages |

Code Block 6: Transform images

At the end processed images are saved in proper directories, related with their class of defects. 10% of images are saved in validation directories, which will be used to checking of accuracy of network model (*Code Block 7).*

|  |
| --- |
| **def** saveFilesToDataset(processedImages, defectClasses):  i = 0  **for** img **in** processedImages:  **if** i % 10 == 0:  path = VALIDATION\_PATH + **'/'** + defectClasses[img[1]]  **else**:  path = DATASET\_PATH + **'/'** + defectClasses[img[1]]  imgHandler.saveImageToFile(img[0], path, str(i))  **if** i % 1000 == 0:  **print 'Saved '** + str(i) + **'images'** i += 1 |

Code Block 7: Save pre-processed images

All operations from this chapter, can be saw in *Code Block 8.*

|  |
| --- |
| **print 'Getting defect classes'** defectClassesDict = dbHandler.getDefectClassDefectStringDictionary() **print 'Defect classes got' print 'Getting defects, images and defects counter'** defects, defectImgs, defectsCounter, defectsCounterDict = getMatchedDefects() **print** str(defectsCounter) + **' successfully got'** processedClasses = getClassesSet(defectsCounterDict)  createClassFolders(defectClassesDict, DATASET\_PATH, processedClasses) createClassFolders(defectClassesDict, VALIDATION\_PATH, processedClasses) processedImages = preprocessData(defects, defectImgs, defectsCounter, processedClasses) **print '... all images successfully processed'  print 'Saving files to dataset...'** saveFilesToDataset(processedImages, defectClassesDict) **print 'Files successfully saved'** |

Code Block 8: Data pre-processing

## Prepared datasets

During performing of thesis, multiple datasets were created. It is related with many attempts of increasing quality of neural network model. Trainings were prepared with different number of classes, different size of images. First trainings were prepared on dataset, where on the images was only ROI of defects. The size of defects were different, so images had to be resized. At the beginning all defects were processed. Unfortunately dataset contains defects with noises, which was caused by resizing. To reduce noise, images with smaller defects than output size were not processed or defects with size close to square were processed. Next attempts were performed on whole images, where defects occur. Initially images were resized 4 times, 2 times according to computing power limitation. After that, trainings on images with original size were performed on the clusters of Prometheus, using plgrid platform. Below are described datasets, which were used to training process, for the needs of this thesis.

* I dataset

First dataset was prepared from all defects, delivered from arcelormittal. There are 33 classes, which contain 21251 defects. Each image contain only ROI, which was resized to 32x32 pixels. This dataset contains following classes: *CGSIS\_Red, Other, Roll\_Mark, Sliver, CGSIS\_Scale, PostCL\_1, R\_Roll\_Mark, Sliver1, CGSIS\_Scratch, Pseudo\_Pattern\_3, Scale\_Comet, Small Coilend, Pseudo\_Scale\_Comet, Scale\_Red, Statistic, Damage, Pseudo\_Shell\_2, Scale\_Red\_3, Unsure\_Track, E\_Crack, Rolled\_In\_Scale\_2, Scale\_Salt\_and\_Pepper, Water, Grind, Rolled\_In\_Scale\_3, Scale\_Stripe, Hole, Rolled\_In\_Scale\_4, Scratch\_Dark, Not\_Classified, Rolled\_in\_Scale\_Sagger, Slag.*

* II dataset

Next dataset limits the number of classes, trying to improve neural network model. Number of classes were limited to 10, and contain 17782 defects. Images contain ROI, with size 32x32 pixels. Following classes were used: *CGSIS\_Scale, R\_Roll\_Mark, Scale\_Salt\_and\_Pepper, Scratch\_Dark, Unsure\_Track, Not\_Classified, Scale\_Comet, Scale\_Stripe, Small, Water.*

* III dataset

Just like in II dataset, only number of classes were limited to 5. These contain 14327 defects, and each has size 32x32 px. In this dataset following classes can be found: *Not\_Classified, R\_Roll\_Mark, Scale\_Comet, Small, Water.*

* IV dataset

Next step to improve results was skipping images, which size of ROI is smaller than output size. This is related with limitation of noises, formed during up resizing of images. Output size of defect was 32x32 pixels, so all records which size has width or height below 32 pixels was rejected. Dataset contain 10 classes, which contain 9957 defects. Following classes was used: *CGSIS\_Scale, R\_Roll\_Mark, Scale\_Red, Scale\_Stripe, Small, Not\_Classified, Scale\_Comet, Scale\_Salt\_and\_Pepper, Scratch\_Dark, Water.*

* V dataset

Fifth dataset contains data, similarly like in fourth dataset with limited number of classes. This dataset has 5 classes, with 9957 defects. This classes were used: *Not\_Classified, R\_Roll\_Mark, Scale\_Salt\_and\_Pepper, Water Small.*

* VI dataset

This is last dataset, where training was done on the transformed images from ROI. For this training only defects with size close to squares were taken. All images with width or height grater twice than second parameter were dropped. Dataset contains 5 classes, with 3925 defects. Following classes were used: *Not\_Classified, R\_Roll\_Mark, Scale\_Salt\_and\_Pepper, Water, Small.*

* VII dataset

This dataset was prepared for training on whole image. Unfortunately, images had to be resized because of computer power limitation. Images were resized from 780x128 pixels to 390x64 pixels. Additionally some records had to be skipped due to computer power limitation, and finally only 6500 defects were used. Dataset contains 5 classes, including: *R\_Roll\_Mark, Scale\_Comet, Scale\_Salt\_and\_Pepper, Small, Water.*

* VIII dataset

Next attempt with whole images were prepared on images, which size was reduced four times, from based size 780x128 pixels to 195x32 pixels. In this case, more defects can be used to training process. Dataset contains same classes like in VII dataset, and 12985 defects.

Size reducing for dataset VIII and IX was prepared on the same dataset using *SizeReduce* python script, which can be saw in *Code Block 9.*

|  |
| --- |
| newHeight = int(HEIGHT \* SCALE) newWidth = int(WIDTH \* SCALE) skipped = 0 processed = 0 **for** subdir, dirs, files **in** os.walk(ROOT\_DIRECTORY):  **for** file **in** files:  path = subdir + **'/'** + file  newSubdir = subdir.replace(ROOT\_DIRECTORY, TARGET\_DIRECTORY)  **if not** os.path.exists(newSubdir):  os.makedirs(newSubdir)  img = imgHandler.openImage(subdir + **'/'** + file)  height, width, channels = img.shape  **if** height != HEIGHT **or** width != WIDTH:  skipped += 1  **continue** img = imgTransformator.resize(img, (newWidth, newHeight))  processed += 1  imgHandler.saveImageToFile(img, newSubdir, str(processed)) |

Code Block 9: SizeReduce.py

This script take the data from directory, and resize each image by the provided scale. It was simpler and faster to create one dataset, and resize it, instead of creating each dataset separately.

* IX dataset

It is a first dataset, which contains images with original size 780x128 pixels. Each next dataset, contains whole images, with this size. Dataset contains 5 classes, with 14327 defects. There was used same classes as in III dataset.

* X dataset

Next dataset was created from dataset IX. In this dataset defects from the end of the band was dropped. These records characterized by black strip on the part of image. This black strip can make the noise in data, and it could be confused for the neural network. Example of image from the end of the band can be saw on Figure 29. Dataset contains 10529 defects.



Figure 29: Image from the end of the band

* XI dataset

Eleventh dataset contains images, which contains only a single defect. Images with 2 or more defects were dropped. Like in previous datasets, this contain 5 classes. Images from the end of the band were dropped too, just like in X dataset. This dataset contains 7523 defects, and following classes are included: *Scale\_Red, Scale\_Stripe, Small, Water, R\_Roll\_Mark.*

* XII dataset

Last datasets contains custom classes. Custom class is a general class in database, which aggregate few classes. This dataset contains 5 custom classes, including: *Periodic, Red, Scale, Scratch, Sliver,* which contain 9778 defects. Dataset contains only a single defects, without defects from the end of the band.

* XIII and XIV dataset

Unfortunately amount of classes was irregular, and two classes contains thousands of defects, while three classes contains only hundreds of defects. For this reason XII dataset was splitted for two separate datasets. Dataset XIII contains 2 classes: *Periodic* and *Scale*, with 7896 defects. Dataset XIV contains 3 classes: *Red, Scratch, Sliver* with 1882 defects.

## Convolutional Neural Network

Having prepared datasets, system can start the training process. Process is supported by *Keras*, which is a high-level neural networks API for python. As a input of neural networks are datasets, prepared by data pre-processing module.

To train defects classifier, Convolutional Neural Network was used. In system, 3 convolution layers network was used, with *relu* as activation function, and *MaxPooling2D* as a pooling layer at the end of each convolution layer. After the last convolution layer, the output was flatted, and then *Dense* layer was used. *Dense* layer allow, that every node in the layer is connected to every node in the preceding layer. *Dense* layer was used two times, at the end of the *flatten* layer, and after dropout of *dense*. CNN model can be saw in *Code Block 10*.

|  |
| --- |
| **def** build(self, width, height, depth, nClasses):  model = Sequential()  inputShape = (height, width, depth)   model.add(  Conv2D(self.CONV\_FILTERS[FIRST], self.CONV\_KERNEL\_SIZE[FIRST], padding=self.PADDING, activation=self.ACTIVATION,  input\_shape=inputShape))  model.add(Conv2D(self.CONV\_FILTERS[FIRST], self.CONV\_KERNEL\_SIZE[FIRST], activation=self.ACTIVATION))  model.add(MaxPooling2D(pool\_size=self.MAX\_POOLING\_SIZE[FIRST]))  model.add(Dropout(self.DROPOUT\_RATIO))   model.add(Conv2D(self.CONV\_FILTERS[SECOND], self.CONV\_KERNEL\_SIZE[SECOND], padding=self.PADDING,  activation=self.ACTIVATION))  model.add(Conv2D(self.CONV\_FILTERS[SECOND], self.CONV\_KERNEL\_SIZE[SECOND], activation=self.ACTIVATION))  model.add(MaxPooling2D(pool\_size=self.MAX\_POOLING\_SIZE[SECOND]))  model.add(Dropout(self.DROPOUT\_RATIO))   model.add(Conv2D(self.CONV\_FILTERS[THIRD], self.CONV\_KERNEL\_SIZE[THIRD], padding=self.PADDING,  activation=self.ACTIVATION))  model.add(Conv2D(self.CONV\_FILTERS[THIRD], self.CONV\_KERNEL\_SIZE[THIRD], activation=self.ACTIVATION))  model.add(MaxPooling2D(pool\_size=self.MAX\_POOLING\_SIZE[THIRD]))  model.add(Dropout(self.DROPOUT\_RATIO))   model.add(Flatten())  model.add(Dense(self.DENSE, activation=self.ACTIVATION))  model.add(Dropout(self.DENSE\_DROPOUT\_RATIO))  model.add(Dense(nClasses, activation=self.DENSE\_ACTIVATION))   **return** model |

Code Block 10: NetworkModel.py

Training process starts with the loading of dataset. As a loading dataset, system needs images, labels of classes names, and number of classes. It is realized by method *getImgesAndLabelsInPath* from *ImagesHandler* class (*Code Block 2*). After loading of dataset, we can build network model. To build network model, system needs following parameters: *width, height, depth, number of classes* (*Code Block 10*). *Depth* parameter means color channel of input data. Next, system needs prepare data to form, which is needed by *Keras* API. Training process is realized by *Keras*, and it is fully automated. After training process, output is serialized and saved as graph. Network model and Network weights and are saved. Training process can be saw in *Code Block 11*.

|  |
| --- |
| images, labels, nClasses = imagesHandler.getImgesAndLabelsInPath(DATASET\_PATH) NetworkModel = Model.NetworkModel()  model = NetworkModel.build(width=WIDTH, height=HEIGHT, depth=DEPTH, nClasses=nClasses)  images = np.array(images, dtype=**"float"**) / 255.0 labels = np.array(labels)  (trainX, testX, trainY, testY) = train\_test\_split(images, labels, test\_size=0.25, random\_state=42)  **print 'del images and labels' del** images **del** labels  trainY = to\_categorical(trainY, num\_classes=nClasses) testY = to\_categorical(testY, num\_classes=nClasses)  **print**(**"[INFO] compiling model..."**) model.compile(optimizer=**'rmsprop'**, loss=**'categorical\_crossentropy'**, metrics=[**'accuracy'**])  aug = ImageDataGenerator(rotation\_range=30, width\_shift\_range=0.1,  height\_shift\_range=0.1, shear\_range=0.2, zoom\_range=0.2,  horizontal\_flip=True, fill\_mode=**"nearest"**)  **print**(**"[INFO] training network..."**) H = model.fit\_generator(aug.flow(trainX, trainY, batch\_size=BATCH\_SIZE),  validation\_data=(testX, testY), steps\_per\_epoch=len(trainX) // BATCH\_SIZE,  epochs=EPOCHS, verbose=1)  **print**(**"[INFO] serializing network..."**) model.save(OUTPUT\_MODEL\_PATH)  plt.style.use(**"ggplot"**) plt.figure() N = EPOCHS plt.plot(np.arange(0, N), H.history[**"loss"**], label=**"train\_loss"**) plt.plot(np.arange(0, N), H.history[**"acc"**], label=**"train\_acc"**) plt.title(**"Training Loss and Accuracy on first approach"**) plt.xlabel(**"Epoch #"**) plt.ylabel(**"Loss/Accuracy"**) plt.legend(loc=**"lower left"**) plt.savefig(OUTPUT\_PLOT)  model.save\_weights(OUTPUT\_MODEL\_WEIGHTS) |

Code Block 11: TrainNetwork.py

# Results of trainings

# Bibliography

1. Szymon Lechwar, (2013), Opracowanie Wizyjnego Klasyfikatora Wad Powierzchni Związanych Z Występowaniem Zgorzeliny W Procesie Walcowania Blach Na Gorąco, Rozprawa Doktorska: AGH University of Science and Technology
2. Parsytec-AG, (2005), HTS-H 4.3 SP1 - Defects Catalogue, Hot Rolling, Aachen.
3. Breitschuh, W., Crowley, G., Dallemagne, L., Deléglise, A., Diaz-Alvarez, J., Valcarcel, J.M., Di Fant, M., Fiori, S., Hemmerlin, M., Koschack, U., Schroyens, K., (2007),ArcelorMittal internal defects catalogue: ArcelorMittal documentation
4. Sun, W., (2005), A study on the characteristics of oxide scale in hot rolling of steel, Wollongong: University of Wollongong
5. Sun, W., Tieu, A. K., Jiang, Z., Lu, C., (2004), High temperature oxide scale characteristics of low carbon steel in hot rolling, Journal of Materials Processing Technology, 155– 156, 1307–1312
6. Sun, W., Tieu, A. K., Jiang, Z., Lu, C., Zhu, H., (2003), Surface characteristics of oxide scale in hot strip rolling, Journal of Materials Processing Technology, 140 (1-3), 76– 83.
7. Verein-Deutscher-Eisenhuttenleute, (1996), Surface Defects in Hot Rolled Flat Steel Products: Verlag Stahleisen mbH
8. Lanteri, V., Thomas, C., Bocquet, H., Yamamoto, H., Araya, S., (1998), Proceedings of the 7th International Conference on Steel Rolling, 423, Chiba
9. Fukagawa, T., Okada, H., Meahara, Y., (1994), Mechanism of red scale defect formation in Si-added hot-rolled steel sheets, ISIJ International, 34 (11), 906-911.
10. Duda, R., Hart, P., Stork, D., (1996), Pattern Classification and Scene Analysis: Part I Pattern Classification: John Wiley & Sons, Inc
11. Hilton, G., Sejnowski, T., (1999), Unsupervised Learning: Fundation of Neural Computation, Cambridge: MIT Press
12. Webb, A., (2002), Statistical Pattern Recognition (2nd Edition), Chichester, England: John Wiley & Sons, Ltd.
13. Ogiela, M. R., (2001), Syntaktyczne metody rozpoznawania obrazów i ich wykorzystanie w analizie wybranych obrazów medycznych, Kraków: Wydawnictwo AGH.
14. Tadeusiewicz, R., Flasiński, M., (1991), Rozpoznawanie obrazów, Warszawa: PWN.
15. Fix, E., Hodges, J., L.J., (1951), Discriminatory analysis — nonparametric discrimination: Consistency properties, Project 21-49-004, Report No. 4, USAF, School of Aviation Medicine, 261–279.
16. Bishop, C., (2006), Pattern Recognition and Machine Learning, New York: Springer.
17. Robbins, H., Monro, S., (1951), A stochastic appriximation method, Ann. Math., 22.
18. Ajzerman, M. A., Brawerman, E. M., Rozonoer, L. I., (1976), Rozpoznawanie obrazów - metoda funkcji potencjalnych, Warszawa: WNT.
19. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., (1984), Classification and Regression Trees, Belmond: Wadsworth
20. Quinlan, J. R., (1979), Discovering rules from large collections of examples: a case study, Expert Systems in the Micro-electronic Age, Edinburgh University Press, Edinburgh, UK, 168–201
21. Quinlan, J. R., (1993), C4.5: Programs for Machine Learning, San Mateo: Morgan Kaufmann
22. Cichosz, P., (2000), Systemy uczące się, Warszawa: WNT.
23. Friedman, J. H., (1999a), Greedy function approximation: A gradient boosting machine: IMS 1999 Reitz Lecture
24. Friedman, J. H., (1999b), Stochastic gradient boosting: Stanford University.
25. Andersen, T., Martinez, T. R., (1995), A provably convergent dynamic training method for multilayer perceptron networks, Proceedings of the 2nd International Symposium on Neuroinformatics and Neurocomputers, 77–84.
26. Schiffmann, W., Joost, M., Werner, R., (1993), Comparison of optimized backpropagation algorithms, European Symposium on Artificial Neural Networks (ESANN), Brussels, Belgium, 97–104
27. Schӧlkopf, B., Smola, A. J., (2000), Learning with Kernels, Cambridge: The MIT Press.
28. Kuhn, H. W., Tucker, A. W., (1951), Nonlinear programming, Proceedings of 2nd Berkeley Symposium, 481-492.
29. Burges, C., (1998), A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery (2), 121-167.