Semantic segmentation of aerial hyperspectral images based on deep learning methods

Filip Mitrev

Abstract

Following the ubiquitous success of deep neural networks, in this project we propose convolutional approach for solving the task of semantic segmentation on aerial hyperspectral imagery. Main issues when dealing with this topic are explained and the problem is deeply researched. A review of existing related papers is given, detailing state of the art approaches. Furthermore, a solution that extracts features with convolutional autoencoder is implemented. Specifically, we will evaluate our method on land cover segmentation from satellite images. Our proposal improves or performs similarly to state of the art proposals in most datasets in the satellite remote sensing domain.

Index Terms

semantic segmentation, remote sensing, precision agriculture, PCA, hyperspectral imaging, deep learning, convolutional autoencoder, pixel classification, SVM

I. Introduction

THE quick emergence of many technologies in the near past, most notably cheap UAVs(Unmanned Aerial Vehicles) and deep neural networks paved the way for processing aerial images and precision agriculture on a scale unseen before. In the context of precision farming, very high resolution data on both spatial and spectral sense will be available, further requiring development of novel algorithms for processing it. In this work we take this into consideration and we propose a system based on a combination of classical and deep learning methods to the problem of land cover segmentation in satellite images. The effects from being able the monitor a field and act accordingly from image analysis can have a great, positive impact. Nowadays, being able to mount a camera on a cheap UAV can save a lot of time, financial and human resources, as well as improve performances. If implemented successfully, this has really high potential in the future, as shown by [1]. Monitoring fields from aerial perspective is our main motivation behind this project and we truly believe that precision agriculture is the future of this industry.

A. Deep Learning

Although deep learning methods have been around for a lot of time in theory, in practice they were virtually non-existent because of the computational power needed to train those kinds of networks. Nowadays, with the advances in computing, most notably shifting the computations from traditional CPUs (Central processing unit) to GPUs (Graphics Processing Unit) and the constant development of more powerful hardware, training very deep neural networks is possible. GPUs excel in parallelization of the processes; they has hundreds or thousands of cores instead of just 4 or 8 in modern CPU. Therefore, GPUs are more convenient for the convolution operation or matrix multiplication, which is one of the most important operations in this type of NN (Neural Networks). Since proposing the use of deep, convolutional neural networks on the ImageNet image classification competition [2] (regarded as the Olympics of machine learning [3]), results improved by a wide margin. The main advantage of this DL (Deep Learning) uprising in machine learning is being able to extract deeper, more abstract features in comparison with the traditional approaches. Furthermore, it has the power of generalization; DL methods are applicable in a variety of problems. Deep neural networks perform better than classical approaches on many tasks in the computer vision field of research and the revolution is set to continue. Extracting deeper features from the input data is bound to produce better results because of the deeper level of abstraction, beating shallow features in nearly every task. The only drawback can be training time, which can take up to weeks or months in some cases. It is dependent on how deep the architecture is, the nature of the input data and the power of the hardware which is doing the training. The winner of 2015 ImageNet classification task [4] used a very complex architecture, 152 layers deep and encapsulating inception models, or network in network. It is believed that the error rate of his model, 3.57% in top-5, is similar or lower than the average human capabilities, suggesting a milestone in machine learning [5]. Another recent milestone worthy of note is the AlphaGo system, which learnt the complicated puzzle game of Go, beating one of the best players in the world in 2016 [6] . Today, frameworks and APIs for the sole purpose of deep machine learning are designed [7], as well as special GPUs and computers for training these complex neural networks [8] [9]. Areas and industries benefiting from this chain of events are computer vision, handwriting and speech recognition, natural language processing, bioinformatics, agriculture, surveillance and autonomous driving, to name a few.

Author: Filip Mitrev, mitrevf@gmail.com
Advisor: Ph.D Daniel Ponsa, Advanced Driver Assistance Systems, Computer Vision Center
Thesis dissertation submitted: September 2016

B. Aerial image analysis

Aerial imaging is the practice of taking pictures of the ground from an elevated position for different purposes, or informally from birds eye point of view. The benefits from such an image are cleaner perspective and covering more space per pixel. Many objects can act as a platform carrying the sensor, including aircrafts, helicopters, UAVs, balloons, rockets, pigeons, kites, airships and parachutes. Aerial images by themselves are perspectively distorted, so to be able to use them as maps and to calculate distances, they have to be orthorectified. Such rectified photos are called orthophotos or orthoimages and with merging more of them, orthophotomaps are created [11]. Aerial images altered in this way represent the data correctly and are useful to the observer. Orthomaps used for war means (reconnaissance) were previously recorded with MAVs (Manned Aerial Vehicles). Obviously, that requires vehicle large enough to accommodate the pilot and the sensor. Usually, airplanes or helicopters were used for this type of imaging. On the other hand, recent advances in technology contributed to the popularization of UAVs. They are smaller, able to carry different sensors and do not require pilot inside the cabin, which translates to low cost, cheap flights and human life is not exposed [Fig.1].

Fig. 1: A typical rotatory wing quadcopter [12]

A satellite is a type of UAV that operates on a higher altitude. It can be powered by solar cells. They are used to map entire areas. Different projects exist that provide these kind of images, but usually they are not publicly available and the price of operating them is beyond the reach of small companies. However, the project Copernicus [14] has changed the perspective and free satellite data of remarkable resolution (Sentinel-2) [13] will soon be available for precision farming tasks. Depending on the equipped sensor, satellites can take images in different spatial and spectral resolution [15]. A few known satellites are ASTER, AVIRIS, ROSIS, ESA Sentinel project, SPOT, MeteoSat, GeoEye and ImageSat. We will focus our work on AVIRIS and ROSIS recordings as they offer the most popular datasets in the remote sensing field, enabling us to easily compare our

performance with regard to the state of the art approaches.

For the purpose of taking images, vehicles are equipped with digital cameras. Cameras on aerial vehicles can have different spatial and spectral resolution. Spatial resolution refers to the number of pixels the sensor is able to record on a single 2D image (height x width), while spectral resolution represents the depth of every pixel; it measures how accurately we can explain the reflectance of the given pixel in the electromagnetic spectrum, with recorded values for the number of spectral bands. We, as humans, usually perceive light in three different bands (red, green, blue), capturing light in wavelengths ranging from 400nm to 700nm [Fig.2]. Wavelengths below 400nm are labelled as ultraviolet and over 700nm is the infrared region of the spectrum.

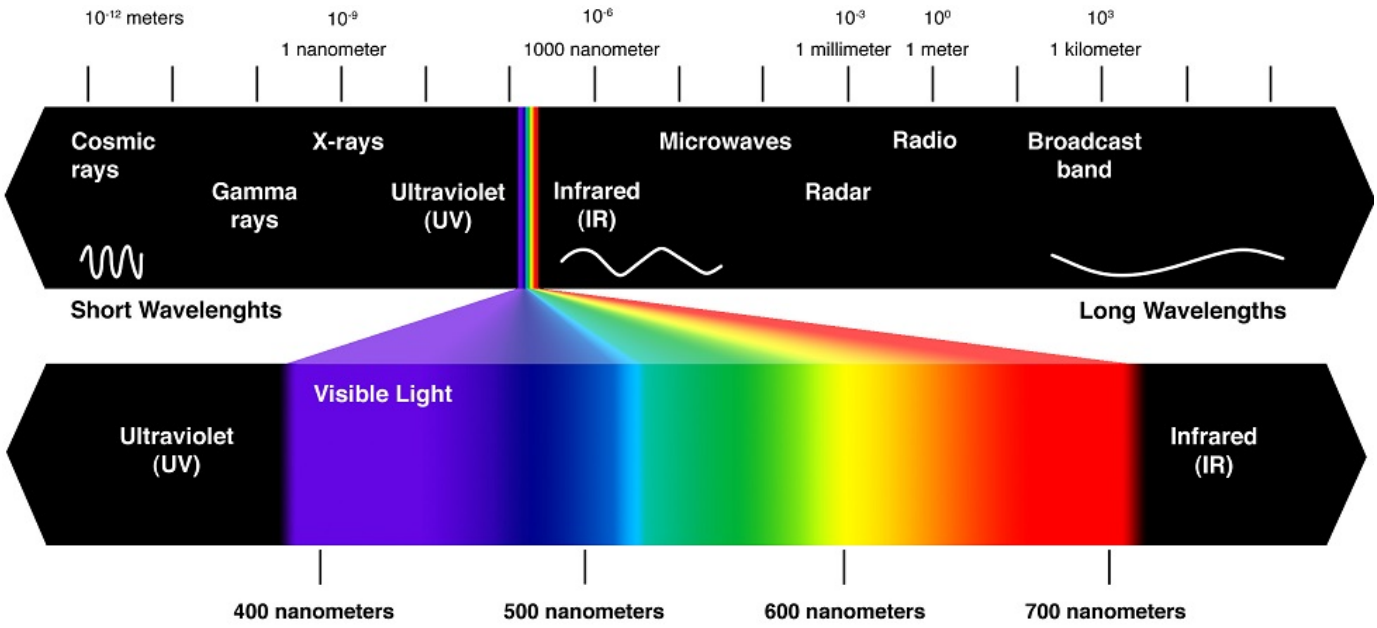

Fig. 2: Electromagnetic spectrum with detail to visible light [16]

Multispectral and hyperspectral images are images taken by special cameras that are able to record and extract more precise information across the electromagnetic spectrum. That enables them to record the light in many narrower spectral bands instead of the usually used three, as described previously. It should be noted that the difference between multispectral and hyperspectral images is just in the sensitivity; the number of bands in which the sensor divides the light. Multispectral images range from several to tens of bands, while hyperspectral images might have up to hundreds of spectral bands. Precisely knowing the spectrum of every pixel is useful in order to find objects, identify materials or detect phases of plants in the agriculture industry. Processing hyperspectral aerial images has a lot of potential for developing precision farming systems. However, there are some relevant issues that have to be dealt with:

- Hardware and other equipment are expensive and rare, which means that datasets can be very hard to find or record. In fact, HS (hyperspectral) imaging is used a lot in military purposes, so it might be confidental. Researchers are constrained to work on existing datasets available to the public. Recently, there has been an effort to make more data available free of charge [17] [13].
- The very high spectral and spatial resolution of some HS images. Processing images with high spatial resolution in addition to high spectral resolution (spectral resolution can range from 102 to 224 bands in some cases) can require a high computational demand to operate on. Having 224 dimensions for every pixel of the image means that one will suffer from the ”curse of dimensionality” [18], making it at least out of reach from conventional computational power by todays standards.
- Even with considerable amount of data, there persists another issue; data is usually partly labeled or not labeled at all. Most of the sucessful machine learning algorithms are based on training with labeled data and labeling every pixel by hand requires a lot of human effort and is therefore expensive. There is a lack of annotated datasets on the satellite imagery domain, forcing researchers to implement systems based on less data.
- Spatial variability of spectral signature. As noted before in [20], pixel-level segmentation without spatial information is doomed, so we must add the neighborhood of the pixel in the feature. On the other hand, materials belonging to the same class can be really sparsely represented on an image and pixels with similar spectral patterns might not be related spatially, a natural occurrence that could pose problems on the classifier.

C. Definition of the problem

Agriculture has been a main industrial branch since ancient times. With the growing concerns of overpopulation and recent ecological issues, the need for sustainable agriculture and maximizing the yield from farms is even bigger. Precision agriculture, or satellite farming is the concept of responding to farming based on observing and measuring the crops. Eventually, the goal of precision agriculture is developing a *decision Support System* which will be able to manage farming in such a way that it optimizes returns given some input data. A well-designed system should maximize profits for the farmer by maximizing harvest yields while minimizing negative impacts on the crops, such as negating diseases with certain chemical [21]. In the described fashion farmers are able to manage and improve many tasks in agriculture, such as observing plant growth and detecting mature crops, quality evaluation, soil monitoring, crop grading and sorting, counting crop instances, land cover segmentation for weed localization, measuring water and nitrogen stress and detecting different diseases [22]. Moreover, precision agriculture offers insight into environmental protection (limiting the use of unnecessary nitrogen levels, preserving water from excessive watering, abandoning fuel powered machines for less air pollution) and economics (lowering the prices in general, competitiveness on the market with optimal management of resources).

In our work, we will focus on using hyperspectral aerial imagery for precision agriculture. Our goal is to achieve land cover segmentation. Semantic segmentation is a subdiscipline of image processing and computer vision. It is the task performed on images to add labels pixel-wise. Or in other words, classify the semantic meaning of every pixel in the image. It is commonly referenced as pixel classification and is also synonymous with scene labeling, pixel-level labeling and pixel-wise classification [23]. So, the goal here is to map every pixel of the image to a label that belongs to a set of existing classes [Eq. 1]:

$$I(x_i, y_j) = c_x, \qquad c_x \epsilon \{c_1, c_2, c_3 ... c_n\} \tag{1}$$

where with 'n' is denoted the number of possible classes, 'i' and 'j' are the width and height of the current pixel respectively, or the spatial location of the processed pixel and 'c' is a class.

Precisely segmented data is valuable knowledge and is helpful for many of the aforementioned tasks in agriculture. Segmented orthomaps behave as decision makers in precision agriculture systems and are followed by taking an action. We can conclude that land cover segmentation is one of the most sough-after branches in this field and although it is partially feasible with normal RGB images, multispectral/hyperspectral data can really contribute to more accurate mapping. The hypothesis here is that every material reflects light in a different way, a property which can be exploited more accurately from images with higher spectral resolution.

Furthermore, we should distinguish between semantic and instance segmentation when segmenting images. The difference here [Fig.3] is evident: in semantic segmentation we only care to which class the pixel belongs, while in instance segmentation we are also interested which instance of that class the pixel represents [24].

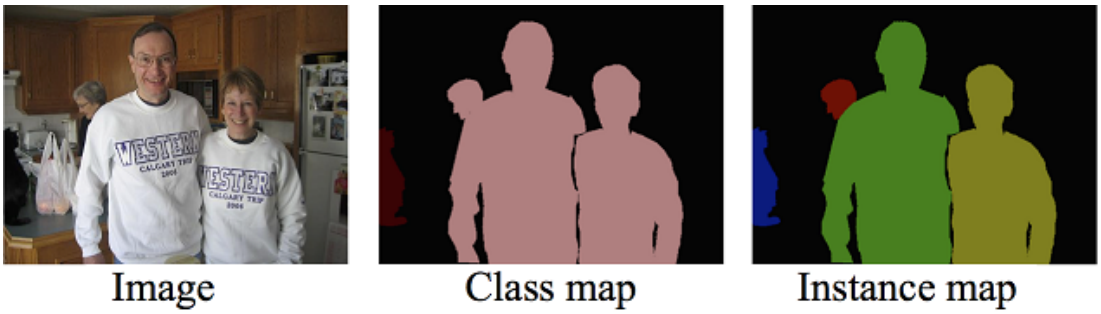

Fig. 3: Semantic vs Instance segmentation [25]

II. Related work

In the next section we review the work done previously by other authors and spot the key findings in their work. We identify state of the art architectures and their basic chronological framework. Moreover, we present the most popular datasets and try to explain why it is hard to compare the results of papers on a universal scale.

A. Most used hyperspectral datasets

Due to the difficulties obtaining hyperspectral datasets for training and testing, most of the papers are constrained to using the ones which are publicly available. However, because of the high spectral resolution of the data and the fact that in some datasets there are uninformative bands, some datasets are available in the original and also corrected form (Salinas, SalinasA and Indian Pines datasets). In the corrected scenes, the bands covering the water absorption are removed beforehand. Many

works cite their results on the following datasets listed in [TABLE I] with their key properties. It should be noted that SalinasA is a subset of Salinas, covering just a small part of the scene. All details on the specific classes, removed bands and land coverage per pixel can be seen in [26]. Mainly, Indian Pines and Salinas datasets contain data of individual crops and vegetation states in 16 classes, while Pavia Centre and Pavia University describe an urban scenario in 9 classes. Although it does not offer hyperspectral segmentation evaluation, lately the challenging dataset COCO (Common Objects in Context) is the state of the art in segmentation testing [27].

Name of dataset	Recorded by	Spectral bands	Spatial resolution	Number of classes
Indian Pines	AVIRIS	224	145x145	16
Salinas	AVIRIS	224	512x217	16
SalinasA	AVIRIS	224	86x83	6
Pavia Centre	ROSIS	102	1096x1096	9
Pavia University	ROSIS	103	610x610	9

TABLE I: Most commonly used hyperspectral datasets with their properties [26]

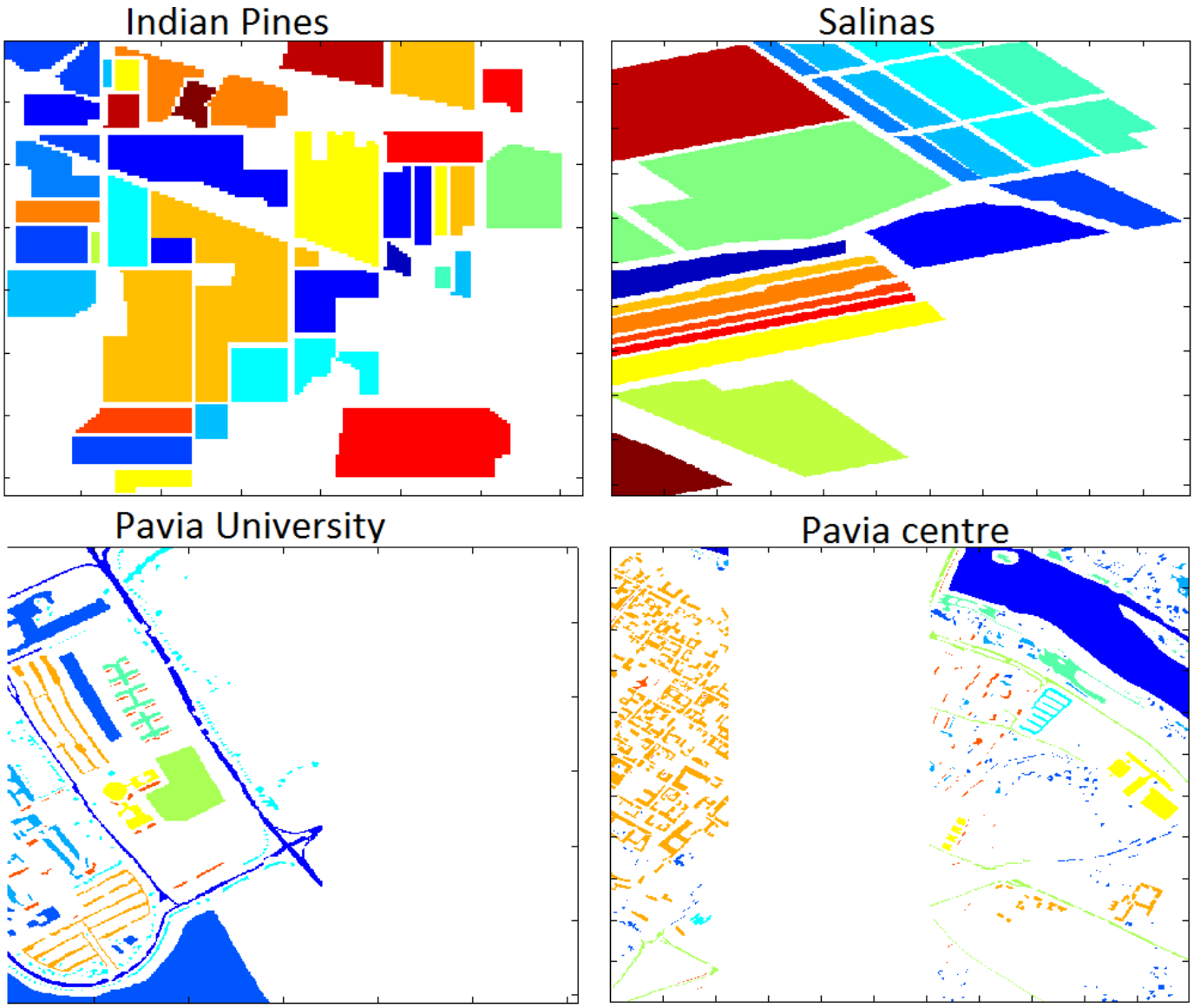

Fig. 4: Ground truth for the most popular hyperspectral datasets

B. Basic framework for image segmentation on hyperspecral data

Based on reviewed bibliography, we are able to present a common pipeline, which is loosely followed by most of the related work on the field [Fig. 5]. According to [10], there are three types of learning, which are related to the amount of data we have available:

- *Purely supervised;* suitable when there are lots of labeled data
- *Layer-wise unsupervised+supervised classifier;* good when labeled data is scarce but there is a lot of unlabeled data
- *Layer-wise unsupervised+supervised backpropagation;* useful when the learning problem is very difficult

The problem we are covering in this work clearly belongs to the second type, which is justifying the presented framework here. In this section the basic steps required to train and test a feasible solution are shown. We would like to note that not every article is strictly following the proposed framework here.

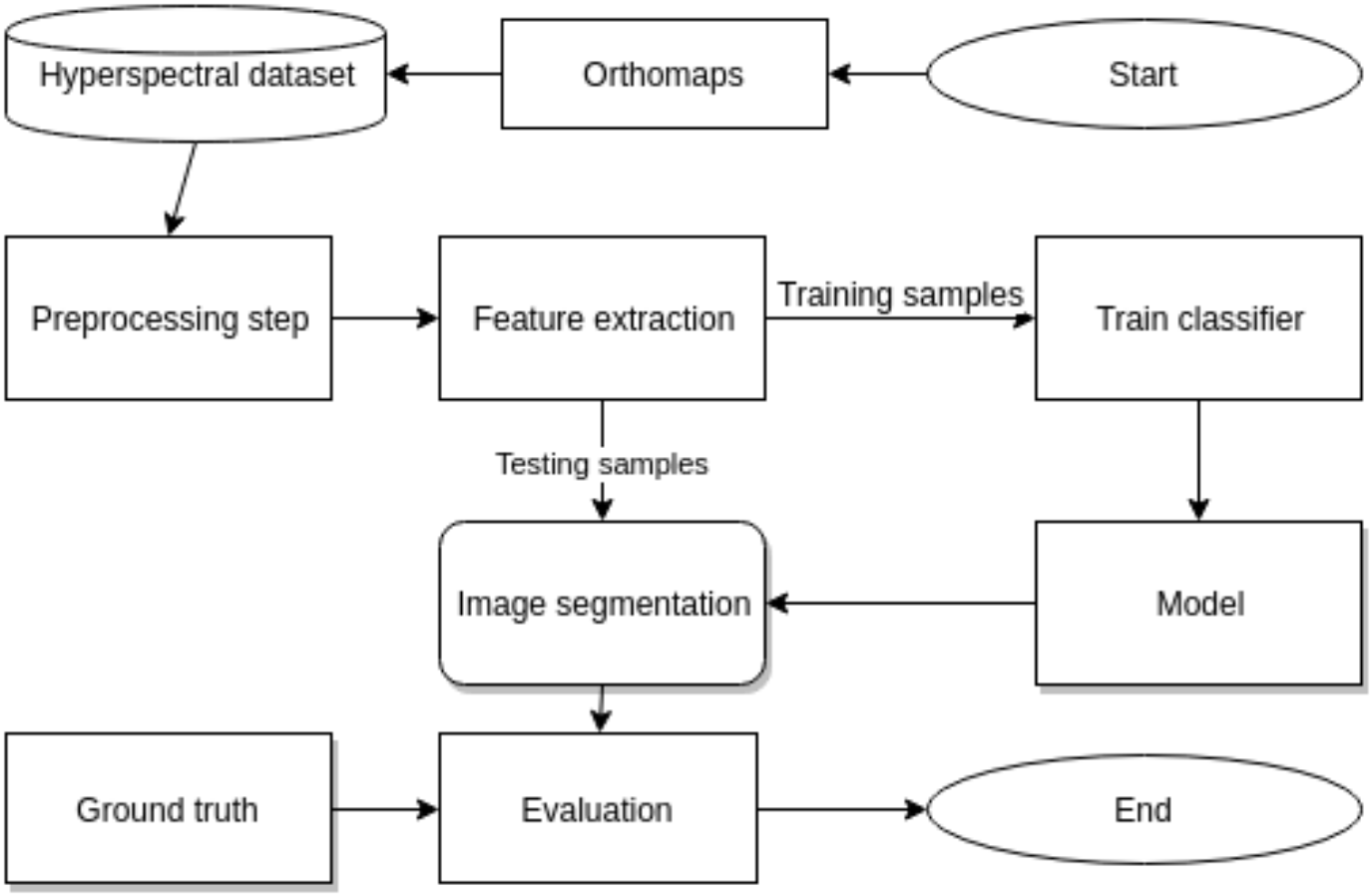

Fig. 5: Commonly used pipeline for semantic segmentation on hyperspectral images

1) Preprocessing step: One of the biggest and most important problems when dealing with hyperspectral data is the dimensionality. The input to the designed architecture is equal to the spatial resolution multiplied by the depth of the image, or the number of spectral bands (width x height x bands). In order to cimcumvent this issue, many authors propose dimensionality reduction. This step can be performed in different ways, but all of them just partially solve the problem, because considerable amount of data is lost in the process. The idea here is to keep only the most relevant pieces of data. Among others, for this step varieties of PCA (Principle Component Analysis) is mostly used as a straightforward way to reduce dimensionalty to a desired level [31] [29]. However, neutral networks like deep autoencoders are emerging as non-linear, more efficient approach to the problem [30]. There is no uniform rule for dimensionality reduction and every author is implementing this step using different meta-parameters in the process [Fig. 6]. This leads to complications when comparing success of similar models, because obtained results are not comparable between each other.

Moreover, when working with neural networks, the input is usually given as a fixed-size image. If our image dimensions differ from the networks requirements, the image should be altered. It can either be stretched to the width and height required or cropped centrally, leaving only the middle exposed.

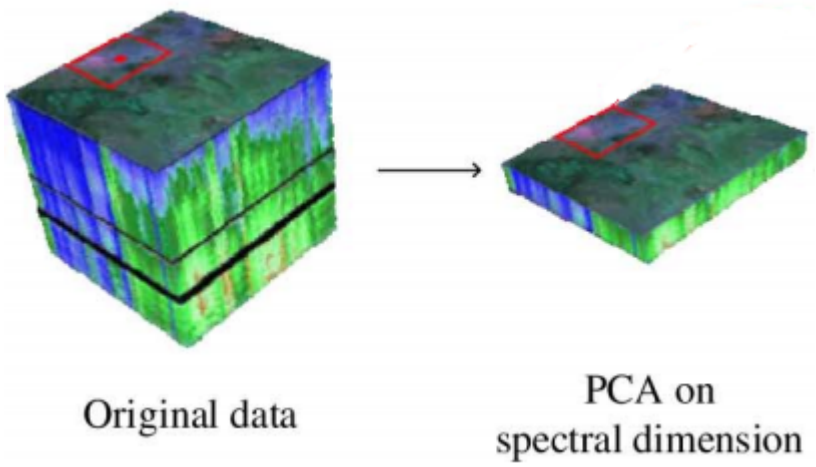

Fig. 6: An example of dimensionality reduction with PCA performed on the spectral dimension [20]

2) Feature extraction: Feature extraction is the process of finding useful ways to interpret the original data. It starts from an initial set of measured data and builds derived values called features, intended to be informative. Feature extraction is closely related to dimensionality reduction. Often the original data is too large for processing directly with algorithm as a input and we look for ways to transform it and select only the most relevant information in a descriptor. The descriptor is believed to be the best representation of that piece of data we could obtain within the required size. Afterwards it is fed to the algorithm on behalf of original data for further processing. Neural networks are implemented for this task, especially autoencoders [33]. Autoencoder is a special type of neural network, capable of mapping the input in as little as possible data and then trying to replicate the input from the encoded data. It practically does encoding and decoding of the data, hence the name. Many variants of autoencoders are present [32]. With stacking more instances of the hidden mapping layer, a deep autoencoder is created. In order to classify pixels, it makes sense to use information of the surrounding of the pixel. That is why a convolutional autoencoder is a logical option for this problem [48]. The idea of using convolutional autoencoder for semantic segmentation, as shown in [49], is very promising. The main reason behind that is the fact that convolutional autoencoder is appropriate for continuous signals (images), unlike the normal autoencoder neural network which does not take into account the spatial component.

3) Classifier training: Before the appearance of deep learning approaches, best performances in land cover segmentation were obtained by SVMs (Support vector machines) [36] and many different variants of them, using different kernels (linear or non-linear). Despite the fact that SVM with non-linear RBF kernel proves to be very effective and robust, when we value ease of implementation and computational cost a linear SVM can be more appropriate. For easier problems it is still the preffered method, beating neural networks on training time. Moreover, they are thought to perform as well as 3-layers deep neural network. Other classical machine learning algorithms and techniques could also be implemented with limited success like decision trees, random forests, linear regression and Adaboost.

Recently, supervised deep learning methods have overtaken the conventional approaches for classification. Although many types of supervised deep neural networks exist, CNNs (Convolutional Neural networks) stand out. They take advantage of the convolution operation. More convolutional layers can be stacked on top of each other for extracting deeper and more abstract features. CNNs can also share weights. Because of that, the number of parameters in comparison to the traditional, feed-forward, fully connected neural networks is smaller. In between the convolutional layers, pooling(usually maxpooling) and layers for non-linearity (Rectified Linear Unit-ReLU) are implemented. There are many techniques which improve CNNs even further, like batch normalization, dropout, weights initialization and meta-parameters to optimize (learning rate, stride, size and number of filters, padding, type of pooling, weight decay) [**?**]. Convolutional Neural networks became the golden standard and the go-to type of network for processing images. Compared to other image classification algorithms, convolutional neural networks use relatively little pre-processing. This means that the network is responsible for learning the filters that in traditional algorithms were hand-engineered. The lack of dependence on prior knowledge and human effort in designing features is a major advantage for CNNs [34].

C. Evaluation metrics

All of the datasets mentioned in [TABLE I] are given with ground truth image. By comparing the label of the ground-truth image to the output of the implemented model for every pixel in the dataset, we can measure how good the model is at classifying. Accuracy [Eq. 2] is the most widely used metric for evaluation in this field of research. In this case, it is simply the ratio between correctly classified pixels and all pixels available.

$$Accuracy = \frac{T}{T+F} \tag{2}$$

However, some authors argue that for segmentation, more general and robust statistic is needed, the Cohens Kappa coefficient. The reasoning behind this is that kappa coefficient takes into account the yielded result occurring by chance [28]. In [Eq. 3], p_0 is the relative observed agreement among raters and p_e is the hypothetical probability of agreement by chance, using the observed data to calculate the probabilities of each observer randomly saying each category. A κ value of 0 means that the classification system is as good as randomly choosing a class, a negative value is returned when the classification is worse than this random choice, and a value of 1 means perfect agreement between raters [35].

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e} \tag{3}$$

III. Our Model

Our approach is generally based on [Fig. 5]. We propose a land cover segmentation system based on applying PCA in the preprocessing step, then using an autoencoder neural network for feature extraction and training a SVM for the pixel classification. The novelty in our proposal is the use of convolutional autoencoder(CAE) for describing pixel regions and SVM on top of it for classification, which to our knowledge have not been previously evaluated in this application domain. In our experimental work we test how fine-tuning the meta-parameters affects the final results, specifically monitoring the effect of the CAE neural network on the whole architecture. In simple terms, we use the dimensionality reduction(PCA)-feature extraction(CAE)-classification(SVM) layout for conducting our experiment. In the following subsections we devise a step-by-step chronological walkthrough for presenting our method.

A. Preprocessing step

The first step is reducing the spectral resolution with Princal Component Analysis (PCA). PCA is useful statistical technique that finds its application in image compression and is also used in finding patterns in data of high dimension [31]. Simply put, PCA is transformation that tries to present the data in a different, more efficient way. Data transformed with PCA is converted to linearly uncorrelated variables called principal components, sorted by importance. With choosing the first 'n' components, we are able to represent our data with less variables. Using this technique, we can efficiently retain the valuable information and variance in the data while minimizing the loss in the process. We select the amount of data we want to keep either as a minimum percentage of the original data or directly choosing the 'n' parameter (first components by importance). The result we get from multiplying them with the original data is the desired representation of the data. This preprocessing step is really important in our model, because working with hundreds of dimensions (bands) will eventually take a lot of space and processing time. Despite the loss of information, it was observed that the data preprocessed with PCA not only reduced the processing time, but surprisingly, can also produce better results in the classification task when compared to the raw data. In spite of the nature of hyperspectral images as highly dimensional data, most of the published papers use PCA in the preprocessing step, although with different parameters.

B. Feature extraction

In order to extract features from the data, we chose to train an autoencoder neural network. There are many variants of autoencoders: convolutional AE, variational AE, fully connected AE, denoising AE, sparse AE, stacked AE, to name a few. We chose using deep CAE for our model, because it makes use of the spatial characteristic with the convolutional layers. This variant have shown good performance on different tasks [48] . One of the main advantages of this approach is the idea of unsupervised learning, which means that training can be done on any data and we do not need labels for each pixel for this step. Our idea is to train the neural network to extract useful features from every sample, while limiting the size of the final descriptor to at most SIFT-like length. It is achieved by stacking more layers which gradually reduce the input data with minimal losses to each layer; in [37] is shown a concrete example of a model and the compression scale of the data is seen. We will evaluate the performance of this method and use it as a ground to base our own model.

The network is trained with stochastic gradient descent technique with Mean Squared Error (MSE) criterion. Some parameters for the training need to be tuned: the database serving as input, number of epochs, number of iterations, batch size and learning rate. Most of them are largely dependent on the amount of data serving as input to the network.

After the training is done, even with relatively small training error we are not guaranteed successful reconstruction. It is needed to check the actual performance with a complete forward pass. The dataset designed for testing with 20% of the samples, previously created with the first step, is loaded and the wanted number of samples is processed with feeding the samples to the saved network. To easily compare the images, both the original image and the reconstructed image (passed through the forward step) are saved in separate files. If the images really resemble semantically, then the trained CAE is considered successful.

Having trained an efficient autoencoder, the samples have to be passed through it in order to encode them. First, we cut off the second part of the network, the decoding (reconstruction) layers. We extract the features from the samples with the shrinking layers, so the last hidden (latent) layer is the output of the encoded sample and we do not need the reconstruction layers. We feed the network all the samples and write the acquired descriptors in a file. Separate matrices for training and testing are built, so the data never interferes between each other. Furthermore, for the following step (classification) the model needs labeled data for training, so we also keep the class of each sample in a separate file accordingly.

C. Multiclass classification

When the samples are encoded and matrices are ready, we are able to train a classifier for multiclass classification problem. In our model we are using SVMs for this task [38]. The matrices with the data as well as the appropriate labels matrix for both training and testing sets are loaded and the data is scaled and centered (whitened), which makes the training converge faster. A one-against-all type of classifier is trained, with feeding the dataset for training (80% of the data we have). One-against-all means that we end up with one classifier for each class represented in the ground truth. Used time is measured for the entire training as well. More about the meta-parameters used for the training is covered in the next section.

Trained classifiers for each class are followed by prediction; testing is done only on new data, never used before in training the autoencoder or SVM classifier (20% of all data). With prediction we get probability estimates for all classes for each classifier. Then the final prediction is made with choosing the maximum value from the probability estimates, saving the values in a separate array. The final predictions are compared with the actual labels. The ratio between them describes how successful our model is - the accuracy measure [Eq. 2]. Accuracy of the model is the simplest and well-known metric. However, we also measure the Kappa Index described before [28], which is perceived as the more precise metric because it negates correctly predicted samples by random occurence. Furthermore, confusion matrix is prepared in order to check samples belonging to which classes are misclassified by the model most often.

IV. Experiments & Results

In this section we are presenting the parameters we used for the model explained in the previous section. We conducted our experiments in different environments; for preprocessing the image we run a script in MATLAB [39]; after dimensionality reduction with PCA, patches of the whole dataset are also cropped in MATLAB. For creation of the database in 'lmdb' [40] format, we use a bash script. Then we shift to the Torch [41], a library designed for deep learning. It is useful because it can efficiently use the GPU for computations, using the cuDNN library from the CUDA API. Torch scripts are written in Lua programming language. Our deep convolutional autoencoder network is trained, tested and run in this environment, using the 'nn' package for neural networks besides the main 'torch' package. The layout of the CAE we use is based on [37]. Finally, the MATLAB version of the library 'libsvm' [38] has been used to implement the classification, using different versions of kernels to train the classifier. For the training of CAEs in Torch, we used a CUDA-enabled mobile low-range GPU (Nvidia 840M with 2GB RAM) with cudnn libraries, which takes advantage of multi-core parallel processing and trains the network faster then CPU. For fitting of SVM classifier in MATLAB we used a low-end mobile CPU (Intel i5-5200U with 2.2 GHz frequency, up to 2.7 GHz with turbo boost technology). The system we experimented on has 8 GB of RAM and uses Linux Mint operative system. In the next section we are presenting the results we achieved in our experiments.

A. Image preprocessing study

The first step, PCA transformation of the data, is straightforward. We opted for using just the first three bands for all datasets and then did all of the experiments on the transformed data, because from the TABLE II is obvious that 3 bands were enough to represent most of the data accurately in all datasets with acceptable losses. The criteria here was not to leave out any single component with significance of more than 1% to the whole set. In the emphasised column the total percentage of data kept is shown. It is worth mentioning that most of the papers use different number of bands as a PCA parameter or percentage of data covered. In the state of the art paper on supervised hyperspectral semantic segmentation [43], they used 10 or 30 bands to keep 99.9% of information. Moreover, [20] in a similar model to ours uses up to 8. Because the reduced hyperspectral datasets after PCA have 3 bands, we can plot them as standard RGB images in which the spectral differences/similarities are visible [Fig. 7] [Fig. 8] [Fig. 9] [Fig. 10] [Fig. 11].

PCA	Amount of information in band, sorted by importance in %					**Total % of info kept**	Original bands	Spatial resolution
	1	**2**	**3**	4	5			
Indian pines (corrected)	**68.4938**	**23.5314**	**1.4964**	0.8215	0.695	**93.5216**	200	145x145
Salinas (corrected)	**74.4737**	**23.5313**	**1.1341**	0.5404	0.1736	**99.1391**	204	512x217
SalinasA (corrected)	**92.0061**	**6.5846**	**0.5882**	0.3601	0.284	**99.1789**	204	83x86
Pavia centre	**70.2403**	**26.0705**	**2.8133**	0.3186	0.1195	**99.1241**	102	1096x715
Pavia University	**58.3181**	**36.1007**	**4.4376**	0.3008	0.2098	**98.8564**	103	610x340

TABLE II: Explanation of vector importance and components kept after PCA for the datasets

Fig. 7: Indian Pines after PCA

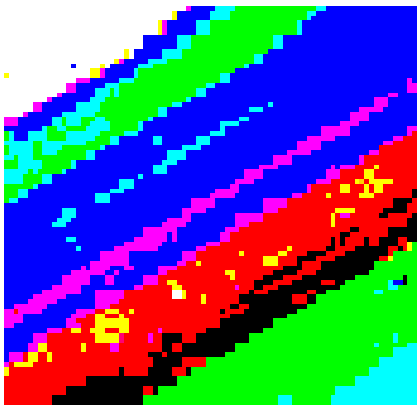

Fig. 8: SalinasA after PCA

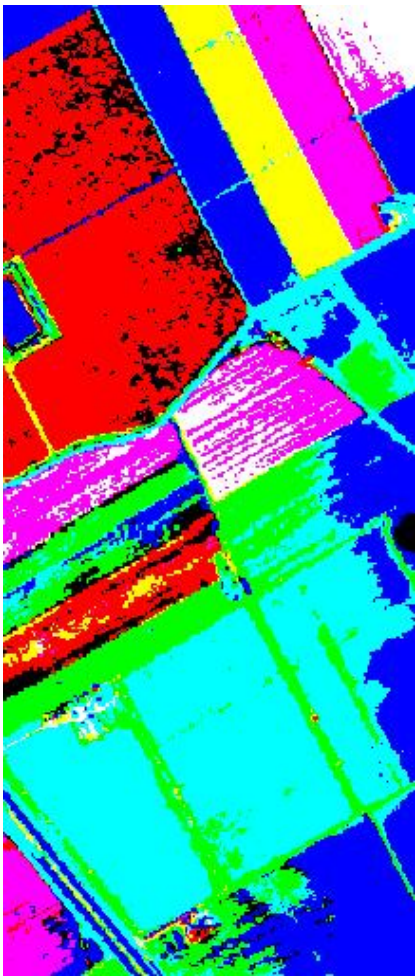

Fig. 9: Salinas after PCA

Fig. 10: Pavia University after PCA

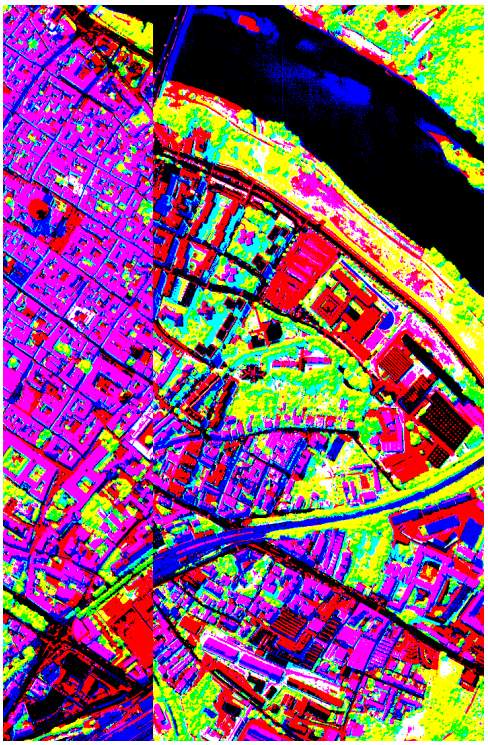

Fig. 11: Pavia Centre after PCA

B. Analysis of CAE layouts

Samples are extracted from the whole dataset only if the pixel in question is labeled (which is around half the pixels in some datasets). Because we take the surrounding of the said pixel, we must crop a patch which has some spatial resolution. We experimented with samples of 16x16 and 64x64 pixels, centered around the labeled pixel. 64x64 proved to be too big region compared to the whole resolution of the image for nearly all datasets, especially because it takes more time to train the autoencoder network. Namely, the size of the trained model accepting 64x64 inputs was 162.5 MB in size, while the layout having 16x16 samples was at most 2.7 MB. So, for the results we are showing, we used 16x16x3 data samples as input to the autoencoder (3 being the spectral resolution of the sample after PCA).

Then our experiment migrates to Torch where we have built a convolutional autoencoder neural network and execute Lua scripts for training and testing the CAE. We experimented with different architectures of convolutional encoder, the number of filters for each convolutional layer, the max pooling layers and the size of the last hidden layer. We tested 3 different layouts. We used the same parameters for all convolutional layers: 3x3 kernel with stride and padding equal to 1 in both directions. These parameters are convenient because they output the data in the same form received at input from the previous layer. Also, for the pooling layer we use maxpooling with 2x2 windows in all occasions.

The first model, [Fig. 12], is compressing the sample to descriptor of length 96, taking advantage of using two pooling and inverse unpooling layers, opposed to one for each in the other two models. It also has the most filters in the last convolutional layer with 24. After quick evaluation it became evident that the performances of this model are inferior, so we discard this model without testing it extensively and focus on testing the following 2 models in our experiments.

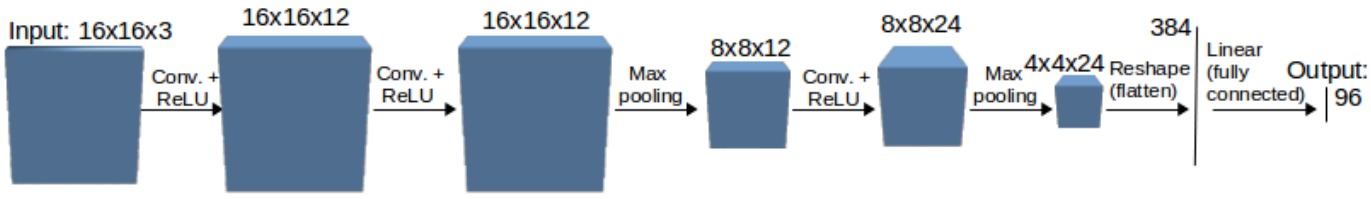

Fig. 12: CAE layout of model 1 (discarded)

The second model,[Fig. 13], is evolved from [Fig. 12]. As shown, we succeeded in reducing the complexity of the network by deleting one pooling layer and the corresponding unpooling layer. To cope with the bigger resolution we are getting in the last convolutional layer of the network (8x8 instead of 4x4 with two pooling layers), we are forced to reduce the third dimension (number of filters from the conv. Layer) to 12. 8x8x12 is flattened to a single array of 768 and is linearly connected to the last hidden layer of 128 neurons, from where the reconstruction begins, using the same layers in inverse order.

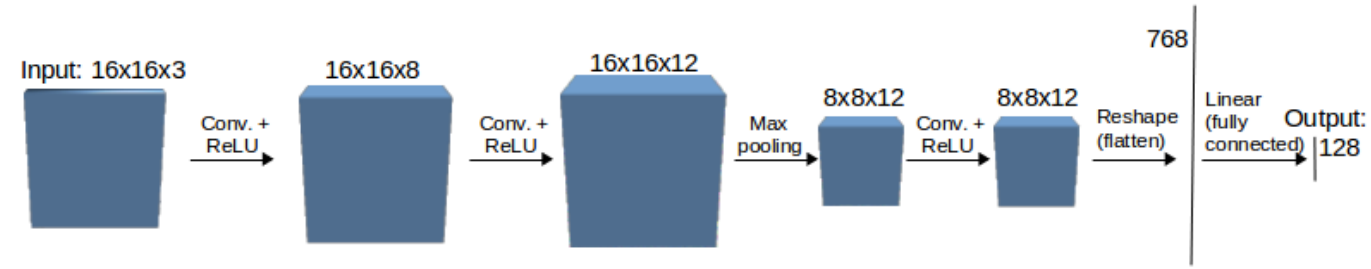

Fig. 13: CAE layout of model 2

For the third layout,[Fig. 14], we only slightly modified [Fig. 13]. There is one small change; it is a bit heavier, increasing the number of filters for the last convolutional layer from 12 to 16. This has increased the size of the trained CAE for 18.5% (2.2mb to 2.7mb).

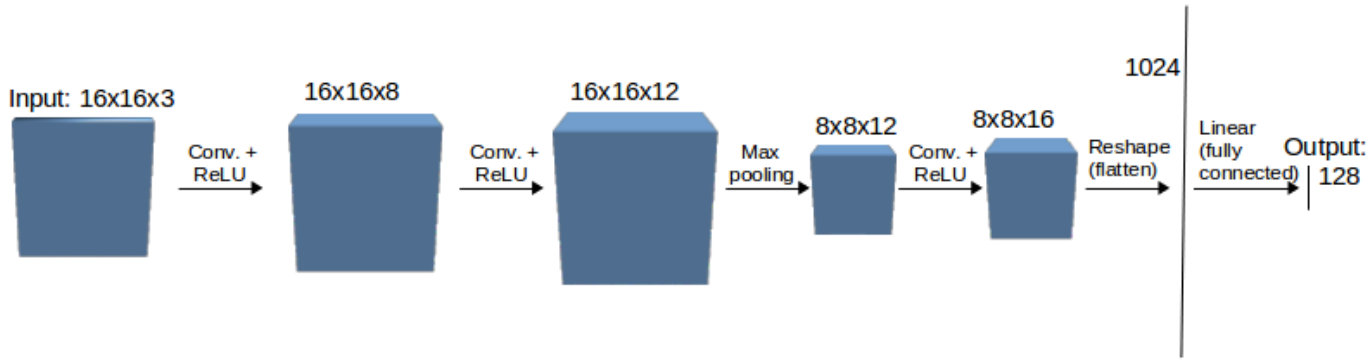

Fig. 14: CAE layout of model 3

All CAEs were trained with stochastic gradient descent method with a mean squared error criterion, while the parameters for learning rate, number of epochs, iterations in each epoch and batch size differ from the amount of data that is fed to the CAE for training and are fine-tuned for each dataset separately.

C. Data compression analysis

With the fixed PCA and CAE output, we can easily calculate the compression level of our system. The original, corrected datasets have between 102 and 204 bands. With the PCA transformation, we keep only 3 channels, which means we compress the data with rate of 97% with 102 bands or 98.5% with 204 bands. Furthermore, with the use of 16x16 patches, every sample is mapped on 16x16x3 before the forward pass in the network. Flattened, it is an array of length 768. Afterwards, the neural network outputs the descriptor of the sample in array of length 128, or compressing the sample additionally with 83% while extracting the final features. Overall, we used 204 times smaller descriptors then the original data for the ROSIS images and 408 times less data for the AVIRIS recordings.

D. Segmentation results

After removing the unnecessary layers (the reconstruction part of the network), data is fed to the CAE and acquired descriptors serve as input to the classifier. We use [38] in MATLAB for SVM classification. There are many different options to choose from for training multiclass SVMs, from which we experimented with different kernel types [42] :

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$
- Radial Basis Function (RBF): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^d)$
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Default parameters are γ=1/number of features, d=3 and r=0. As shown above, we have constrained to using 128 features. Although the default kernel in usage is RBF, our experiments have shown that polynomial kernel is on par or in some cases, better than RBF kernel. Linear and Sigmoid kernels perform a lot worse and take more time to train, but nonetheless are included in the results from the experiments.

In this section we are presenting results from segmentation on all available datasets with features extracted from CAE trained separately for each dataset using the model2[Fig. 13] layout for CAE. Parameters used, time consumed, number of samples and final training errors for every individual CAE are visible in TABLE III.

Autoencoders trained								
	Epochs	Iterations	Batch size	Learning rate	Train samples	Test samples	Training time	Training error
Indian Pines	30	57	128	0.01	7294	1823	475	0.027882
Salinas	30	158	256	0.01	40580	10144	3049	0.005502
SalinasA	20	30	128	0.01	2830	707	219	0.017964
Pavia University	30	122	256	0.01	31212	7803	2488	0.011333
Pavia centre	30	224	512	0.01	115040	28760	8355	0.005092

TABLE III: Convolutional Autoencoders trained for every dataset

Using these networks, we are able to compress the samples and then reconstruct them from the last hidden layer. Examples of original input patches and reconstructed images via the network are shown in [Fig. 15]. Decoding is not needed for the task of semantic segmentation, it is done in order to check the validity of the CAE. Having low training error does not warrant good performance of the network, even more because we do not employ a validation set to prevent overfitting.

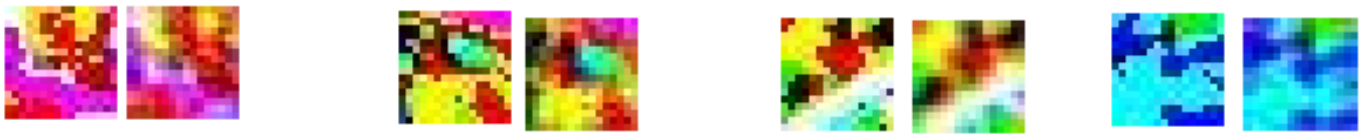

Fig. 15: Examples of original and reconstructed images

Training data is then fed to the corresponding network and both training and testing sets are processed. Afterwards, multiclass SVM model is fitted on the training data and classification is done. Results for each dataset with their own classifier are presented in the tables (TABLE IV, TABLE V, TABLE VI, TABLE VII, TABLE VIII.

Indian Pines			
7294 train/1823 test samples. Training time: 475 sec.			
30 ep./57 iter/128 bs/ 0.01 lr | training err: 0.027882			
Kernel	Accuracy	Kappa index	Training time
Linear	0.6742	0.6213	592
Polynomial	0.9967	0.9962	206
RBF	0.9967	0.9962	189
Sigmoid	0.2847	0.0833	175

TABLE IV: CAE details and segmentation results on Indian Pines using model 1

Salinas			
40580 train/10144 test samples.Training time: 3049 sec.			
30 ep./158 iter/256 bs/ 0.01 lr | training err: 0.005502			
Kernel	Accuracy	Kappa index	Training time
Linear	0.8983	0.8864	3049
Polynomial	0.9216	0.9125	2019
RBF	0.9265	0.9179	2016
Sigmoid	0.5903	0.5364	3855

TABLE V: CAE details and segmentation results on Salinas using model 1

SalinasA			
2830 train/707 test samples. Training time: 219 sec.			
20 ep,30 iter,0.01 lr, 128 bs. Training error: 0.017964			
Kernel	Accuracy	Kappa index	Training time
Linear	1	1	1
Polynomial	1	1	2
RBF	1	1	2
Sigmoid	0.9122	0.8857	29

TABLE VI: CAE details and segmentation results on SalinasA using model 1

Pavia University			
31212 train/7803 test samples. Training time: 2488 sec.			
30ep,122iter,256 bs, 0.01 lr. Training error: 0.011333			
Kernel	Accuracy	Kappa index	Training time
Linear	0.904	0.8758	7379
Polynomial	0.9978	0.9972	1032
RBF	0.9969	0.996	756
Sigmoid	0.613	0.4696	2116

TABLE VII: CAE details and segmentation results on Pavia University using model 1

Pavia centre			
115040 train/28760 test. Training time: 8355 sec.			
30 ep, 224 iter. 512 bs, 0.01 lr. Training error: 0.005092			
Kernel	Accuracy	Kappa index	Training time
Linear	0.9652	0.9506	68028
Polynomial	0.9977	0.9968	12749
RBF	0.997	0.9958	13832
Sigmoid	0.8454	0.775	20318

TABLE VIII: CAE details and segmentation results on Pavia Centre using model 1

The average performance for both metrics (accuracy and Kappa index) for every type of SVM kernel is shown in TABLE IX.

Averages for kernels			
	Accuracy	Kappa index	Training time
Linear	0.88834	0.86682	15809.8
Polynomial	0.98276	0.98054	3201.6
RBF	0.98342	0.98118	3359
Sigmoid	0.64912	0.55	5298.6

TABLE IX: Comparison of SVM kernels (averages on all datasets)

It can be concluded that polynomial and RBF are most successful kernels in our experiment. In favor of cleaner and simpler presentation, we will restrain from showing results recorded with linear and sigmoid SVM kernels.

In the resluts is a noticeable drop in performance on the Salinas dataset is seen. We dug deeper and saw the confusion matrix for this particular dataset, shown in Fig. 16. Most of the misclassified samples in this dataset are belonging to classes 8 and 15. In the ground truth table for Salinas, these classes are labeled as grapes and vineyards. One would expect them to have similar hyperspectral reflectance, even more after PCA transformation when all the spectral information is stored in just 3 channels. In comparison to other literature, we got worse results in this dataset in the expense of less bands used while transforming the data with PCA. In our opinion, these are two very similar plants with different vegetation state (growth). The similarity can be confirmed with comparing the ground truth of the Salinas scene [Fig. 4] with the Salinas image after PCA

16x16 double

	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	389	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	259	8	0	0	0	0	0	0	0	0	0	0	0
5	0	3	518	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	749	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	679	0	0	0	0	0	0	0	0	4
8	0	0	0	0	0	1919	0	0	0	0	0	0	254	0
9	0	0	0	0	0	0	1222	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	608	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	197	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	356	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	169	0	2	0
14	0	0	0	0	0	0	0	0	0	0	0	199	0	0
15	0	0	0	0	0	312	0	0	0	0	0	0	973	0
16	0	0	0	0	33	0	0	0	0	0	0	0	0	192

Fig. 16: Confusion matrix for a Salinas classification test

[Fig. 9], where on the top left side the 2 reddish patches are in fact the misclassified classes. Moreover, these two clusters of pixels are spatially close and with 16x16 receptive field a some of features could have been trained jointly between them.

Due to different amount of data for each dataset, training and testing SVMs also can be computationally expensive. In TABLE X the average training times in relation to the SVM kernel and the dataset are presented.

Kernel/dataset	Indian Pines	Salinas	SalinasA	Pavia Uni.	Pavia Centre
Linear	592	3049	1	7379	68028
Polynomial	206	2019	2	1032	12749
RBF	189	2016	2	756	13832
Sigmoid	175	3855	29	2116	20318

TABLE X: Training times for different types of SVMs

It is apparent that not only Polynomial and RBF SVM models achieve the best results, but they are also converged faster when compared to linear or Sigmoid kernels.

E. Segmentation results using a generic CAE

Having evaluated these results, we were left with little chance to improve. It is known that it gets harder as you approach to 100% success rate. There is evidence that the semantic segmentation problem is not as big as thought before, but to our knowledge, there isn't a more challenging dataset. Our goal for the second part of the experiment was to see if we can train a generic CAE, which can extract useful features from all samples including the ones which belong to other datasets. In other words, how general and robust the features we are getting from the individual CAE are. The motivation and need for this is proving that these CAEs are not constrained to their dataset and can be efficient enough on other data. With that in mind we built a slightly larger deep neural network (not deeper for that matter), using the model3 layout described in Fig. 14. The encoder was trained solely on Indian Pines data. Parameters for the training were 50 epochs with 91 iterations in batches of 100 with learning rate equal to 0.01. The error after training was 0.02029. On top of the training set, SVM is trained with the default parameters as in the usual pipeline. Expectedly, this is the best result we achieved throughout the project for the Indian Pines dataset. With the polynomial kernel, we managed to classify all the samples from the testing data correctly [TABLE XI]. According to our observations, it is due to the usage of model3 [Fig. 14] layout, which performs slightly better compared to model2 [Fig. 13].

We conducted the test with the same autoencoder trained only on the Indian Pines dataset for extracting features from other datasets. Surprisingly, we can say that the success of this CAE translated well into the other datasets too, despite having different classes semantically and even different number of classes overall. Useful features were extracted from all datasets,as shown in TABLE XI, although the recorded performance was a bit worse then in the previous experiment with the individual CAEs for all datasets except Indian Pines. The point here is to see if a bit larger network would outperform the better quality of the data provided with feature extraction of the own dataset, which proved to be a close affair.

	Accuracy		Kappa Index	
	Polynomial	RBF	Polynomial	RBF
Indian Pines	1	0.9978	1	0.9975
Salinas	0.9217	0.9217	0.9125	0.9125
SalinasA	1	1	1	1
Pavia Univer.	0.9953	0.9951	0.9939	0.9937
Pavia Centre	0.997	0.9957	0.9958	0.9939

TABLE XI: Performance with generic CAE trained on Indian Pines data using model3 layout

After recording pleasing results in all datasets except Salinas, we felt there is room for improvement on this particular dataset. A new CAE was trained on the Salinas test set, this time with 100 epochs. It took 8638 seconds to train with an error of 0.0045. Polynomial SVM was fitted on top of the encoded data, resulting in 0.937 accuracy and 0.9296 Kappa index. That bettered the previous results on this dataset by a thin margin.

When training classification models, one of the most important factors is the amount of data fed to the model. This being said, Salinas training set has 40580 labeled samples to train on. On the other hand, the biggest dataset we had available, Pavia Centre, had 115040 samples in its training set respectively. Encouraged by the performance of TABLE XI, we encoded the samples from Salinas using the CAE trained on Pavia Centre and the corresponding SVM achieved *0.9492 accuracy and 0.9362 Kappa* index with polynomial kernel and 0.9414 accuracy and 0.9346 Kappa with RBF SVM, which proved to be the top performance on the Salinas dataset. Conclusion from this is that having more data can be a big advantage, even if the data is with worse quality. Having said that, TABLE III shows that training the Salinas CAE took 3000 seconds, while training the Pavia Centre CAE spent more than 8000 seconds, hence the better results.

F. Comparison of achieved performance with the State of the art

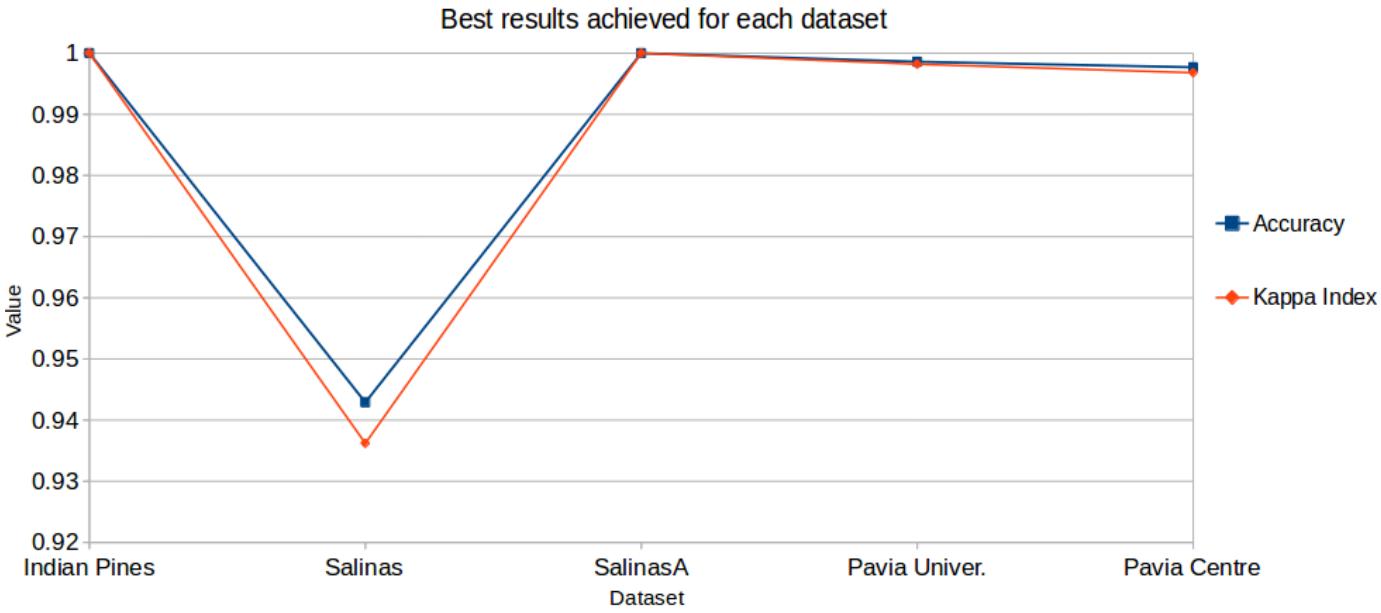

Fig. 17: Best performance recorded on every dataset

Finally, the best results recorded for each dataset are presented in Fig. 17. Despite the slight improvements, our model still underperforms on the Salinas dataset. However, we can say that we are generally pleased with the overall performance. Our model performs as well as or even better then state of the art approaches. As noted before, it is hard to compare the results from different works in the field of hyperspectral semantic segmentation, mainly because all of them use different settings (parameters), hardware equipment (GPUs and CPUs), different depth of models, need different training time and test on different datasets. Nonetheless, we did compare our approach to other results in the field. Results are presented in TABLE XII.

	[46]	[50]	[45]	[44]	[20]	[43]	Our result
Indian Pines	0.76	0.84(κ)	0.9016	0.9208	-	0.9991	**1**
Salinas	-	-	0.926	-	-	**0.9953**	0.9429
Pavia Univer.	-	-	0.9256	-	0.9852	0.9962	**0.9986**
Pavia Centre	-	-	-	-	-	**0.9991**	0.9977

TABLE XII: Comparison to State of the art

In [46] the authors are classifying on the Indian Pines dataset without using neural networks. RBF SVM is tested on data compressed with different techniques, from which JPEG2000 proves to be most successful. They achieve around 0.76 accuracy and 0.74 kappa ratings at best, which is a lot worse then our results. However, they do not learn a compression system but use of the shelf JPEG2000 and learn SVM on top of it.

Our method is most similar to [20], because it follows the same steps (PCA-autoencoder-classificator). Instead of SVM, the authors in the end use learning regression for classification. They record 0.9852 accuracy and 0.9807 Kappa index for the Pavia University dataset, which is a bit worse then our result on this dataset. In spite of that, instead of 128 as we did, they used at most 60 neurons for the last hidden layer.

Another unsupervised method for feature extraction is [50], where 0.84 Kappa Index is achieved on Indian Pines. But, it is worthy of note that the network used here is simpler(6 layer CNN, 5x5 receptive field) and data is scarce.

In [45], CNN is trained with just 200 samples per class for Indian Pines, Salinas and Pavia University scenes, recording accuracy of 0.9016, 0.9260 and 0.9256 accordingly. Although it is worse than our performance, one must take into regard that it is trained with 23% or less data from the whole set. This is bettered by [44], where with the same data on Indian Pines dataset 0.9206 accuracy is achieved.

Due to the fact that we use deeper network and more data for training, our results clearly outperform the ones mentioned above. However, [43] has better ratings in some datasets. Authors make use of a CNN for exploiting spectral and spatial features and then multi-layer perceptron neural network for the classification task. Firstly, they employ PCA, keeping 99.9% of the original data, which translates to 30 components for Indian Pines and 10 components for all other datasets. They use 5x5 receptive field for every labeled pixel, which a lot smaller then ours, 16x16. According to [43], having receptive field larger then 5x5 no further improves the performance of the model. Furthermore, fields larger than 13x13 affects the performance for the worse.

As explained above, our experiments produced far better results then some papers to which we compare. But, as previously said, it is unfair to compare results because between all models there are differences. The focus for many of the cited work was to build a model capable of segmenting with low amount of data.

V. Conclusions

Overall, we are satisfied by the produced results. Our main objective in this project was to test the convolutional autoencoder as a deep neural network and using it, derive a framework for hyperspectral image segmentation. We proposed a novel solution, using PCA+CAE+SVM for pixel classification. After extensive experimental work has been done, we are in position to confirm our hypothesis that using convolutional variant of the autoencoder neural network for extracting features is an efficient method when used on continious signals like hyperspectral images. Coupled with classifier on top of the encoded samples, this system can be very successful in the land cover segmentation domain, as shown in this work.

An observation point we made is how the autoencoder affects the final segmentation performance and how useful the extracted features are. In other words, how the depth of the network contributes to the quality of the encoded sample and which layers of the CAE morel are most important. Our model is 8-9 layers deep (without reconstruction layers), consisting of convolutional layers, maxpooling and ReLU layers for non-linearity. A common talking point for autoencoder networks, according to many papers, is the unpooling in the reconstruction layers. We tried padding from the sides to avoid unpooling which resulted in bad performance. The number of convolutional layers is also important, but there is a trade-off between performance and deeper network, consequently meaning more parameters to train. After every convolutional layer, we employ a ReLU layer. Convolutional layers themselves can vary by size. We only used 3x3 filters with padding and stride equal to 1, so we end up with the same spatial size in the next layer. The number of filters for these layers is different, but tends to increase as we go deeper (e.g. 381216). Naturally, this comes with the expense of more parameters to tune and larger network. After the results of the generic CAE, we can confirm that the CAE successfully trained on nearly any of the datasets with enough data can be effective at extracting features from other hyperspectral images. This is, in our opinion, similar to the way we can use any of the deep, pretrained networks (VGG, AlexNet, ResNet) with only fine-tuning the last, fully connected layer for the task, because the low-level features are similar in many problems.

We can conclude that we are satisfied with the obtained results. The performance of our system approaches the state of the art in the field, and is even improving upon it in some cases. Particularly encouraging is the comparison with [20], where we outperform it with a similar method (PCA-feature extraction from autoencoder-classificator).

As future lines of research, we think that room for improvement may be changing the criterion function for the training

of the CAE. We used mean squared error as a criterion throughout our project, but there are sources which claim that this criterion function doesn't work well with autoencoders [52]. In future experiments, one might also consider trying other types of SVM for multiclass classification; one vs one instead of one vs all. However, according to reports, although it is faster to train, one vs one SVMs actually perform worse [51]. Another idea for a future work might be trying different types of classifiers or replacing the CAE-SVM method in the pipeline with single deep neural network for classification.

We consider the problem of hyperspectral semantic segmentation nearly solved for individual datasets, but the real issue here is building and training an general, robust model that can segment all the scenes. That is, being able to do multi-class segmentation with number of classes in the 100s, without having the ground truth for a similar environment. Nearly all authors agree on the fact that correctly annotated data is really scarce, hence the process of labeling every pixel by hand. The answer for this may lie in the recent efforts of the Copernicus project. We strongly believe that having more hyperspectral data available for free, promised by the Sentinel-2 mission, will inspire more in-depth research in this field of remote sensing. In near future we will probably oversee numerous applications in the precision agriculture industry.

ACKNOWLEDGMENT

I would like to express immense gratitude to my advisor, Ph.D Daniel Ponsa, for presenting the idea and making it available, for being patient, accessible and guiding me throughout the thesis.

I am also sincerely thankful to my family & friends for their unconditional support and being the main motivation behind this work.

REFERENCES

[1] *Gamaya*, [Online]. Available: http://gamaya.com/. [Accessed 1 6 2016].
[2] A. Krizhevsky, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012
[3] X. Giro, *ConvNet for global recognition*, Barcelona, 2016.
[4] K. He, *Deep residual learning for image recognition*, 2015.
[5] A. Karpathy, Stanford, *Convolutional Neural Networks for Visual Recognition*, lecture 7, 2016. [Online]. Available: http://www.youtube.com/watch?v=AQirPKrAyDg. [Accessed 1 6 2016].
[6] "AlphaGo versus Lee Sedol," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/AlphaGo_versus_Lee_Sedol. [Accessed 1 6 2016].
[7] *Deep learning tools and frameworks*[Online]. Available: http://www.kdnuggets.com/2015/12/deep-learning-tools.html. [Accessed 1 6 2016].
[8] Nvidia, *Specialized deep learning supercomputer* [Online]. Available: http://www.nvidia.com/object/deep-learning-system.html. [Accessed 1 6 2016].
[9] Nvidia, *GPU for deep learning* [Online]. Available: http://www.nvidia.com/object/tesla-p100.html. [Accessed 1 6 2016].
[10] Yann LeCun [Online]. Available: http://yann.lecun.com/. [Accessed 9 9 2016].
[11] Wikipedia, "Orthophotomap," [Online]. Available: https://en.wikipedia.org/wiki/Orthophotomap. [Accessed 4 6 2016].
[12] Quadcopter [Online]. Available: http://s-media-cache-ak0.pinimg.com/736x/8e/25/a4/8e25a451053e52e768d2febb7fa202a0.jpg. [Accessed 4 6 2016].
[13] *Sentinel-2*, Earth observation mission, ESA [Online]. Available: http://en.wikipedia.org/wiki/Sentinel-2 [Accessed 8 9 2016]
[14] *Copernicus*, Earth observation project, ESA [Online]. Available: http://en.wikipedia.org/wiki/Copernicus_Programme [Accessed 8 9 2016]
[15] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Satellite_imagery. [Accessed 4 6 2016].
[16] Electromagnetic spectrum, [Online]. Available: http://www.theskepticsguide.org/wp-content/uploads/2014/12/Visible-spectrum.jpeg. [Accessed 1 6 2016].
[17] NASA, [Online]. Available: http://www.nasa.gov/feature/jpl/nasa-japan-make-aster-earth-data-available-at-no-cost. [Accessed 1 6 2016].
[18] Y. Bengio, Quora, *Curse of dimensionality* [Online]. Available: http://www.quora.com/What-is-the-curse-of-dimensionality. [Accessed 5 6 2016].
[19] P. Dayan, *Unsupervised learning*, The MIT encyclopedia of the cognitive sciences.
[20] C. Yushi, L. Zhouhan, Z. Xing, W. Gang and G. Yanfeng, *Deep Learning-Based Classification of Hyperspectral Data*, IEEE Journal of selected topics in applied earth observations and remote sensing, vol. 7, no. 6, 2014.
[21] Precision Farming, NASA, [Online]. Available: http://earthobservatory.nasa.gov/Features/PrecisionFarming/. [Accessed 4 6 2016].
[22] S. Lalit and A. Leisa, *A survey of image processing techniques for agriculture*, Edith Cowan University , 2014.
[23] M. Thoma, "Stack overflow," [Online]. Available: http://stackoverflow.com/questions/33947823/what-is-semantic-segmentation-compared-to-segmentation-and-scene-labeling. [Accessed 1 6 2016].
[24] J. Johnson, Stanford, *Convolutional Neural Networks for Visual Recognition*, lecture 13, [Online]. Available: http://www.youtube.com/watch?v=Bo5amIoRoUE. [Accessed 1 6 2016].
[25] Class vs instance segmentation, UCLA. [Online]. Available: http://www.stat.ucla.edu/ xiaochen.lian/paspart_challenge/images/header.png. [Accessed 1 6 2016].
[26] *Hyperspectral datasets*, EHU [Online]. Available: http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes. [Accessed 1 6 2016].
[27] *ms COCO*, Microsoft, [Online]. Available: http://mscoco.org/. [Accessed 1 6 2016].
[28] *Cohen's Kappa*, Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Cohen%27s_kappa. [Accessed 1 6 2016].
[29] *PCA visually explained in 3D*, Setosa, [Online]. Available: http://setosa.io/ev/principal-component-analysis/. [Accessed 5 6 2016].
[30] G. Hinton, *Deep autoencoders*, [Online]. Available: https://www.youtube.com/watch?v=c-37A9BHWzg. [Accessed 1 6 2016].
[31] Aaron Hertzmann, David J. Fleet and Marcus Brubaker, *Principal Component Analysis*, Computer Science department, Toronto, 2015 http://www.cs.toronto.edu/ g8acai/teaching/C11/Handouts/PCA.pdf
[32] Ian Goodfellow, Yoshua Bengio, Aaron Courville *Deep learning book*, Available: http://www.deeplearningbook.org/ , 2016.
[33] Stanford, *Autoencoders* [Online] Available: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/ [Accesed 9 9 2016].
[34] Convolutional neural network, Wikipedia. [Online] Available: http://en.wikipedia.org/wiki/Convolutional_neural_network [Accesed 9 9 2016]
[35] E. C. Hidalgo, *A land cover classification system for UAV multispectral imagery*, 2015.
[36] *Support Vector Machines*, CS Toronto. Available: http://www.cs.toronto.edu/ g8acai/teaching/C11/Handouts/SupportVectorMachines.pdf [Accesed 9 9 2016]
[37] Siavash Khallaghi, *Training Autoencoders on ImageNet Using Torch 7*, 2016. Available: http://siavashk.github.io/2016/02/22/autoencoder-imagenet/ [Accesed 9 9 2016]
[38] Chang, Chih-Chung and Lin, Chih-Jen, *LIBSVM: A library for support vector machines*. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm .
[39] *MATLAB* IDE, MathWorks. Available: http://es.mathworks.com/products/matlab/ .
[40] *LMDB format*, Wikipedia. Available: https://en.wikipedia.org/wiki/Lightning_Memory-Mapped_Database .

[41] *Torch* computing framework, Available: http://torch.ch/ .
[42] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, *A Practical Guide to Support Vector Classification*, 2016 Available: http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf /
[43] K. Makantasis et. al. *Deep Supervised learning for hyperspectral data classification through convolutional neural networks*, IGARSS, 2015.
[44] L. Hyongtae and K. Haesung, *Contextual deep CNN based hyperspectral classification*.
[45] Wei Hu, Yangyu Huang,Li Wei, Fan Zhang,Hengchao Li, *Deep Convolutional Neural Networks for Hyperspectral Image Classification*, 2015.
[46] F. Garca-Vlchez, *On the Impact of Lossy Compression on Hyperspectral Image Classification and Unmixing*, 2011.
[47] J. V. D. Weijer, *ConvNets for local recognition*, Barcelona: UAB Masters in Computer Vision, 2016.
[48] Johathan Masci, *Stacked Convolutional Auto-Encoders for Hierachical Feature Extraction*, 2011.
[49] B. Vijay, K. Alex and C. Roberto, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," 2015.
[50] R. Adriana, G. Carlo and C.-V. Gustau, *Unsupervised Deep Feature Extraction for Remote Sensing Image Classification*, 2016.
[51] Jean-Philippe Fauconnier, *SVMtypes*, "One-to-Rest" and "One-to-One". Available: https://www.quora.com/In-multi-class-classification-what-are-pros-and-cons-of-One-to-Rest-and-One-to-One [Accesed 9 9 2916]
[52] Daniel Waterworth, *Whats wrong with autoencoders?*, 2016, Available: https://danielwaterworth.github.io/posts/what's-wrong-with-autoencoders.html [Accesed 9 9 2016]

CONTENTS