UNIVERSITAT POLITÈCNICA DE CATALUNYA
UNIVERSITAT DE BARCELONA
UNIVERSITAT ROVIRA I VIRGILI

Master in Artificial Intelligence

Master of Science Thesis

MY MASTER THESIS TITLE

Hadi Keivan Ekbatani

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
FACULTAT DE MATEMÀTIQUES (UB)
ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA (URV)

Supervisor:
Oriol Pujol Vila
Department of analysis
and applied Mathematics,
Universitat de Barcelona (UB)

Co-supervisor:
Santiago Segui Mesquida
Department of analysis
and applied Mathematics,
Universitat de Barcelona (UB)

February 01, 2016

Acknowledgments

I would like to sincerely thank my supervisors Oriol Pujol and Santi Segui for their support, guidance and mentorship. I greatly appreciate their demanding and inquisitive scientific attitude, while keeping always a calm and positive mindset. They are, in my mind, an inspiring example of what university professors should be.

Furthermore I would also like to dedicate this master thesis to my beloved parents and sister for their unsparing supports. My gratitude knows no bounds.

Another special acknowledgment must be made to my friends in the master: Denis, Jeroni, Philipp, Pablo, Lorenzo, Iosu, Ferran and many others for the endless studying hours, morning coffees after crunching the brutal assignments all night long, valuable and constructive discussions which allowed me carry out this program.

Abstract

NOT YET

Contents

List of Figures

Part I

title

1 Introduction

1.1 Motivations

The concept of learning to count is an important educational/developmental milestone which constitutes the most fundamental idea of mathematics. In Computer Vision[99], the counting problem is the estimation of number of objects in a still image or video frame. Learning to count visual objects is a new approach towards dealing with detecting object in the images and video, which has been recently proffered in the literature[108, 84, 56, 16, 90]. It arises in many real-world applications, including cell counting in microscopic images[36], monitoring crowds in surveillance systems[85, 102], and performing wildlife census or counting the number of trees in an aerial image of a forest[13, 82][67].

Artificial intelligence and computer vision share topics such as pattern recognition and machine learning[72, 73] techniques. Consequently, computer vision is sometimes seen as a part of the artificial intelligence field or the computer science field in general. Recent machine learning methods applied for computer vision tasks, require large number of data for the learning process. To learn to count the object of interest in an image or video, various object features need to be designed, extracted or detected during the learning phase. The complexity of feature detection process in vision tasks, restrict their usage in large-scale computer vision applications thus demanding more efficient solutions to alleviate, expedite and improve this process.

One recent and commonly used method to facilitate feature detection process is application of deep Convolutional Neural Networks(CNN)[95, 57, 58, 91, 53, 96]. One of the promises of deep CNN is replacing handcrafted features with efficient algorithms for unsupervised or semi-supervised feature learning and hierarchical feature extraction[93]. CNNs have been claimed and practically proven to achieve the most assuring performance in different vision benchmark problems concerning feature detection and classification[24, 95, 22].

1.2 Objectives

This work sets forward several objectives:

1. To apply deep CNN for feature detection and classification in a learning to count problem where the concept of interest will be counted by no giving explicit information about what we are counting to the system, except for its multiplicity in the image.

2. To synthetically create datasets of images, as realistic as possible, and completely automatically annotated for CNN to train with.

3. To explore the behavior of proposed algorithm on synthetic datasets of different types comparing the performance with a state-of-the-art outcomes[90].

4. To analyze the performance of designed system on real world crowd counting problem comparing the results with state-of-the-art results[16].

1.3 Contributions

This thesis contains the following contributions:

1. It proposes the problem of object representation as an indirect learning problem casted as learning to count strategy. The devised algorithm is capable of counting the number of pedestrians in the image that does not depend on object detection or feature tracking. The model is also privacy-preserving in a sense that it can be implemented with hardware that does not produce a visual record of the individuals in the scene.

2. It provides a synthetically generated and automatically labeled dataset of pedestrians using unlabeled University of California San Diego(UCSD) pedestrian dataset used in [70], to train a counting deep convolutional neural network which is adequate for apprehending the underlying representations of the learned features. To this end, we describe a counting problem for MNIST dataset to demonstrate the capability of the internal representation of the network for classifying digits with no direct supervising while training.

3. The proposed model is able to count the number of people in the real and unseen dataset using the features learned by training the network on synthetic training set. To our knowledge, this is the first crowd counting system trained by synthetic data that successfully operates continuously on real data.

4. Along with the validation of our proposal in the following ways:

   - First, we learn to count even hand-written digits using MNIST dataset.
   - Second, we validate the system quantitatively on a large synthetic dataset of pedestrian, containing 100,000 images with maximum 30 pedestrians in each image.
   - Last but not least, we count the number of pedestrians in the manually labeled dataset of 3375 images provided by[Chan and Vasconcelos, 2013].

1.4 Organization

This report takes off with the review of Deep Learning(DL) as a branch of Artificial Intelligence (AI) which deep convolutional neural networks belong to, and moves on to introduce a deep CNN's basic architecture and components in details along with the definition of applied optimization methods and hyper-parameters in our work (Section 2.1). This section ends with a review of state of the art (Section 2.2).

In section 4, we describe our proposal for constructing a deep neural network to tackle feature detection issue learning to count problems.

Section 5 revolves around the empirical experiments and analysis along with the peculiarities of proposed data creation process and network modeling.

Lastly, in Chapter 6 we conclude the report with a short summary of the scope of work conducted and the new areas of research that this master thesis has opened.

2 Background and Definitions

In this section, we go over the preliminarily concepts that help understand the contributions of this work. We start by looking at the family of methods to which Deep Convolutional Neural Networks belong, followed by a more detailed look at the method in question better name this 'method in question', explaining the some hyper-parameters incorporated for optimizing the proposed model.i think the explanation of your method should be done in a diff section

2.1 Deep Learning

One of the central challenges of Artificial Intelligence (AI) is solving the tasks that are easy for people to perform but hard for them to describe formally – problems that we solve intuitively, that feel automatic, like recognizing spoken words or faces in images. 'one approach to that challenge' maybe The solution is to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, where each concept is defined in terms of its relation to simpler concepts. This hierarchy of concepts allows the computer to learn complex notions by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, it would be a deep graph with many layers. For this reason, we call this approach *Deep Learning*[43].

Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to perform quickly, can be accomplished via deep learning, given sufficiently large models and datasets of labeled training examples. Other tasks, that can not be described as associating one vector to another, or that are difficult enough such that a person would require time to think and reflect in order to accomplish the task, remain beyond the scope of deep learning for now[43].

In other words, Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving ML closer to one of its original goals: Artificial Intelligence. Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound and text[98].

2.2 Deep Neural Networks

there is a package in latex – don't remember how it's called – that lets you define acronynims and reuse them across the text with an automatic list of acronims being generated – ask pablo A standard neural network (NN) consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Shallow NN-like models with few such stages

have been around for many decades if not centuries[88]. However, theoretical results strongly suggest that in order to learn the kind of complicated functions that can represent high-level abstractions (e.g. in vision, language, and other AI-level tasks), one needs deep architectures. Deep Neural Networks are composed of multiple levels of non-linear operations, such as those present in neural nets with many hidden layers or in complicated propositional formulate re-using many sub-formulate[7]'propositional formulate re-using many sub-formulate' – i dont really understand that.

2.2.1 Back Propagation

Backward Propagation of errors (BP) was the main advance in the 1980's that led to an explosion of interest in NNs. BP is one of the most commonly used methods for training NNs. The idea behind BP is that it repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the network and the desired one. As a result of the weight adjustments, internal *hidden* units come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units[113].

Specifically, BP computes how fast the error changes as we adjust a hidden activity by using error derivatives with respect to hidden activities'what are hidden activities' – it comes up out of the blue. Since each hidden activity can have a notable effect on many output units and consequently on the error, a combination of these effects must be considered. This aggregation is done efficiently which allows us to compute error derivatives for all the hidden units quickly at the same time. Computing the error derivatives for the hidden activities, it would be easy to get the error derivatives for the weights going into a hidden unit which is the key to be able to learn efficiently.

2.2.2 Weight Sharing

Transition missing: smth like 'another itegral part part/building block/technique' of deep learning is 'weight sharing' Weight sharing refers to having several connections controlled by a single parameter (weight). Weight sharing can be interpreted as imposing equality constraints among the connection strengths. An interesting feature of weight sharing is that it can be implemented with very little computational overhead[62]. The weight sharing technique has an interesting side effect of reducing the number of free parameters, thereby the capacity of the machine and improving its generalization ability[64]. how is weight sharing relevant in this research. for instance, 'we will later try to modify weight sharing / capitalize on it to build a more efficient model'

2.3 Convolutional Neural Networks

Convolutional Neural Networks are a specialized kind of neural network for processing data that has a known grid-like topology such as image data which can be thought of as a 2D grid of pixels. CNN are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers[43]. Essentially, CNNs combine three architectural ideas to ensure some degree of shift and distortion invariance of local receptive fields, shared weights (or weight replication), and, sometimes, spatial or temporal sub-sampling[64] what are

'local receptive fields', 'spacial/temporal subsampling'. The following components compose the main body of any CNN architecture:

2.3.1 Convolutional layer

Each unit of a convolutional layer receives inputs from a set of units located in a small neighborhood in the previous layer. With local receptive fields, neurons can extract elementary visual features such as oriented edges, end-points and corners. These features are then combined by the higher layers[64]. In addition, elementary feature detectors that are useful on one part of the image are likely to be useful across the entire image. This knowledge can be applied by forcing a set of units, whose receptive fields are located at different places on the image, to have identical weight vectors[113]. The outputs of such a set of neurons constitutes a *feature map*. At each position, different types of units in various feature maps compute different types of features. A sequential implementation of this, for each feature map, would be to scan the input image with a single neuron that has a local receptive field, and to store the states of this neuron at corresponding locations in the feature map[64].

Units in a feature map are constrained to perform the same operation on different parts ot the image. A convolutional layer is usually composed of several feature maps (with different weight vectors), so that multiple features can be extracted at each location.

2.3.2 Pooling/Sub-sampling layer

Once a feature is detected, its' exact position becomes less important as long as its' approximate position relative to other features is preserved. Furthermore, as the dimensionality of applying a filter is equal to the input dimensionality, we would not be gaining any translation invariance with these additional filters, we would be stuck doing pixel-wise analysis on increasingly abstract features. In order to solve this problem, a *subsampling* layer is introduced.

Subsampling, or down-sampling, refers to reducing the overall size of a signal. In many cases, such as audio compression for music files, subsampling is done simply for size reduction [100]. But in the domain of 2D filter outputs, subsampling can also be thought of as reducing the sensitivity of the output to shifts and distortions. One of the most applied subsampling methods used in [63], is known as 'max pooling'. This involves splitting up the matrix of filter outputs into small non-overlapping grids (the larger the grid, the greater the signal reduction), and taking the maximum value in each grid as the value in the reduced matrix. By applying such a max pooling layer in between convolutional layers, we can increase spatial abstractness as we raise feature abstractness[100].

2.3.3 Activation functions

To go from one layer to the next, a set of units compute a weighted sum of their inputs from the previous layer and pass the result through a non-linear activation function[59]. There are many possible choices for the non-linear activation functions in a multi-layered network, and the choice of activation functions for the hidden units may often be different from that for the output units. This is a consequence of the fact the hidden and output units perform different roles[9].

At present, the most popular non-linear function is the Rectified Linear Units (ReLU), which is simply the half-wave rectifier $f(z) = max(z, 0)$. In the past decades, neural nets used smoother non-linearities, such as $tanh(z)$ or $1/(1 + exp(-z))$, but ReLU typically learns much faster in

networks with many layers, allowing training of a deep supervised network without unsupervised pre-training[59].

The rectifier activation function allows a network to easily obtain sparse representations. For example, after uniform initialization of the weights, around 50% of hidden units continuous output values are real zeros, and this fraction can easily increase with sparsity-including regularization. Apart from being more biologically plausible, sparsity also leads to mathematical advantages. On the other hand, one may hypothesize that the hard saturation at 0 may hurt optimization by blocking gradient back-propagation. However, experimental results done byGlorot et al. suggest that hard zeros can actually help supervised training[42].

2.3.4 Local Response Normalization

ReLUs have the desirable property that they do not require input normalization to prevent them from saturating. If at least some training examples produce a positive input to a ReLU, learning will happen in that neuron. However, we still find that the following local normalization(LRN) scheme aids generalization. This sort of response normalization implements a form of lateral inhibition wtf is that? inspired by the type found in real neurons, creating competition for big activities amongst neuron outputs computed using different kernels[57].

This scheme bears some resemblance to the local contrast normalization scheme proposed by Jarrett et al. in [52] without mean activity subtraction 'mean activity subtraction' – ?? which has led to error rate reduction in [57] and [50].

2.3.5 Fully connected/Inner product layer

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via *fully connected layers*(IP). A fully connected layer takes all neurons in the previous layer (be it fully connected, pooling, or convolutional) and connects it to every single neuron it has. Fully connected layers are not spatially located anymore (you can visualize them as one-dimensional), so there can be no convolutional layers after a fully connected layer.

2.4 Model Optimization

this really does not belong to the background section In this section we briefly describe some optimization methods along with definition of hyper-parameters used in our model.

2.4.1 Stochastic Gradient Descent

It has often been proposed to minimize the *empirical risk* (training set performance measure). For more detailed description, see [104]) using *gradient descent*(GD)[11]. The standard gradient descent algorithm updates the parameters θ of the objective *J(θ)*

$$\theta = \theta - \alpha \bigtriangledown_\theta E[(J(\theta)] \tag{2.1}$$

you need to clarify what each variable stands for in this equation where the expectation in the above equation is approximated by evaluating the cost and gradient over the full training set (Empirical Risk Minimization (ERM)). Stochastic Gradient Descent (SGD) simply does away the expectation in the update and computes the gradient of the parameters using only a single

or a few training examples. The new update is given by, are you sure a comma is needed before the equasion?

$$\theta = \theta - \alpha \bigtriangledown_\theta J(\theta; x^{(i)}, y^{(i)}) \tag{2.2}$$

with a pair $(x^{(i)}, y^{(i)})$ from the training set[77].

Generally, each parameter update in SGD is computed with respect to a few training examples or a mini-batch as opposed to a single example. The reasons for this are twofold[77]:

1. The variance in the parameter update is reduced, potentially leading to a more stable convergence.

2. It allows the computation to take advantage of highly optimized matrix operations that should be used in a well vectorized computation of the cost and gradient. A typical mini-batch size is 256, although the optimal size of the mini-batch can vary for different applications and architectures.

3. One final but important point regarding SGD is the order in which we present the data to the algorithm. If the data is given in some meaningful order, this can bias the gradient and lead to poor convergence. Generally, a good method to avoid this is to randomly shuffle the data prior to each epoch of training.

2.4.2 Weight Decay

As a part of BP algorithm and a subset of regularization methods, *weight decay* adds a penalty term to the error function by multiplying weights to a factor slightly less than 1 after each update.

It has been observed in numerical simulations that a weight decay can improve generalization in a feed-forward neural network. It is proven that a weight decay has two effects in a linear network. Firstly, it suppresses any irrelevant components of the weight vector by choosing the smallest vector that solves the learning problem. Secondly, if the size is chosen right, a weight decay can suppress some of the effects of static noise on the targets, which improves generalization significantly[75].

2.4.3 Momentum

The *momentum* method introduced by [Polyak, 1964] is a first-order optimization method for accelerating gradient descent that accumulates a velocity vector in directions of persistent reduction in the objective across iterations. Given an objective function $f(\theta)$ to be minimized, momentum is given by:

$$\nu_{t+1} = \mu\nu_t - \varepsilon\bigtriangledown \tag{2.3}$$

$$\theta_{t+1} = \theta_t + \nu_{t+1} \tag{2.4}$$

where $\varepsilon > 0$ is the learning rate, $\mu \in [0, 1]$ is the momentum coefficient, and $\bigtriangledown f(\theta_t)$ is the gradient at θ_t [94].

For example, if the objective has a form of a long shallow ravine leading to the optimum and steep walls on the sides, standard SGD will tend to oscillate across the narrow ravine since the negative gradient will point down one of the steep sides rather than along the ravine towards the optimumcitation for that?. The objectives of deep architectures have this form near local optima and thus standard SGD can lead to very slow convergence particularly after the initial

steep gains. Momentum is one method for pushing the objective more quickly along the shallow ravine[77].

3 State of the art review

Counting the number of an object of interest in an image can be approached from two different perspectives, either training an object detector, or training an object counter[90]. In the field of object detection, numerous works have been previously proposed[80, 19, 87, 26, 56, 71, 107]. Most of these research works follow a taxonomy which consists of three paradigms underneath to count the objects:

1. Object detection, which are based on boosting appearance and motion features[108, 107], Bayesian model-based segmentation[117], integrated top-down and bottom-up processing[66, 79][16].

2. Visual feature trajectory clustering. This paradigm counts objects by identifying and tracking visuals over a time period. Feature trajectories with coherent motion are then clustered and the number of clusters is the estimate of the number of moving people[84, 14][16].

3. feature-based regression. These methods usually work by first, subtracting the background, second, measuring various features of the foreground pixels such as total area[80, 26], edge count[19, 87], or texture [71]; and finally estimating the crowd density or crowd count by a regression function, e.g. linear[80, 26], piece-wise linear [87], or neural networks[19, 87].

   In recent years, feature-based regression has also been applied to outdoor scenes. For example, [56] applies neural networks to the histograms of foreground segment areas and edge orientations. [31] estimates the number of people in each foreground segment by matching its shape to a database containing the silhouettes of possible people configurations, but is only applicable when the number of people in each segment is small (empirically, less than 6)[16].

By reason of the fact that almost all the above algorithms detect the whole objects in an image (e.g. whole pedestrians), these methods have moderate performance in very noisy or crowded images with significant occlusion, Wu and Nevatia 114, Lin et al. 68, introduced methods to address this issue. Wu and Nevatia 114 proposed *edgelet features*(an edgelet is a short segment of line or curve) as new type of silhouette oriented features to deal with the problem of detecting individuals in crowded still images. Respectively in [68], Lin et al. used *Accumulated Mosaic Image Difference(AMID)* method to extract crowd areas having irregular motion.

As a similar line of work in the course of object counting and more specifically crowd counting, in [84, 14, 65], different object tracking approaches were taken to detect and count moving objects in the scene. However, the deployment of these vision surveillance technologies are invariably met with skepticism by society at large, given the perception that they could be used to infringe on the individuals' privacy rights. While a number of methods that do not require explicit detection or

tracking have been previously proposed[80, 19, 87, 26, 56, 71, 31], they have not fully established the viability of the privacy-preserving approach[16]. The tension of privacy-preserving is common in all areas of data-mining[101, 106].

In order to tackle privacy preserving issue, Chan et al. 16 presented a novel approach with no explicit object segmentation or tracking to estimate the number of people moving in each direction(towards and away from camera) in a privacy-preserving manner. An outline of the crowd counting system appears in figure 3.1:

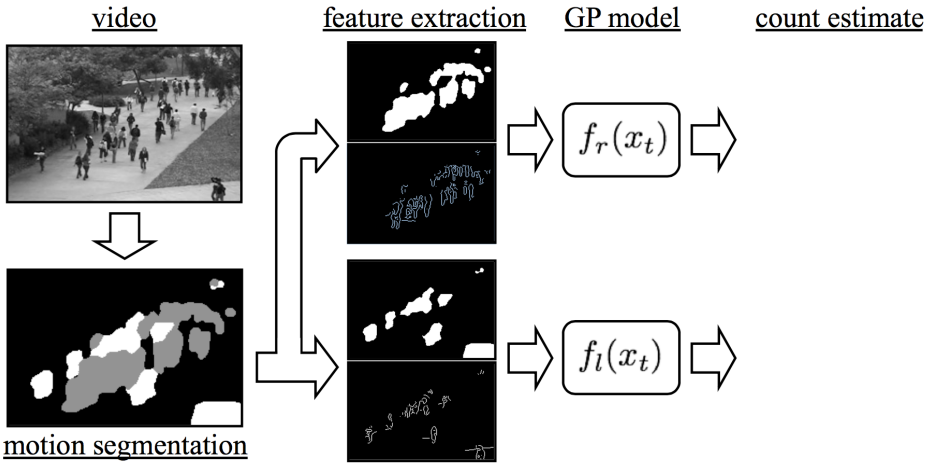

Figure 3.1: Crowd counting system: the scene is segmented into crowds with different motions. Normalized features that account for perspective are extracted from each segment, and the crowd count for each segment is estimated with a Gaussian process[16].

Chan et al. used a mixture of *dynamic textures*[32, 17] to divide the video frames into regions containing moving pedestrians in different directions. When adopting mixture of dynamic textures, the video is represented as collection of spatio-temporal patches which are modeled as independent samples from mixture of dynamic models[32]. The mixture model is learned through Expectation-Maximization(EM) algorithm[17]. Video locations are then scanned sequentially, a patch is extracted at each location, and assigned to the mixture component of largest posterior probability. The location is declared to belong to the segmentation region associated with that component[16]. The resulting segmentations of their work are illustrated in figure 3.2:

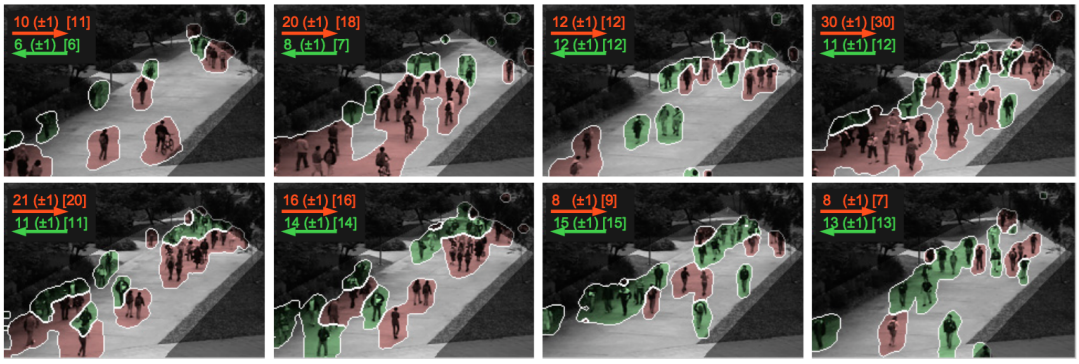

Figure 3.2: Crowd counting results: The red and green segments are the "away" and "towards" crowds. The estimated crowd count for each segment is in the top-left, with the (rounded standard-deviation of the GP) and the [ground-truth]. The Region Of Interest (the area in the walkway in which the pedestrians are counted and labeled) is also highlighted[16].

After segmenting the moving pedestrians, extracting features from the video segments is done at three phases:

- Segment features to capture segment shape and size. Features such as area, perimeter, perimeter edge orientation and perimeter-area ratio.
- Internal edge features contained in a crowd segment are a strong about the number of pedestrians in the segment[26, 56]. For instance, total edge pixels and edge orientation.
- Texture features which are based on gray-level co-occurrence matrix(GLCM) (see [48] for more details) were applied for image patches classification into 5 classes of crowd density in[71]. Due to the task similarity, Chan et al. adopted a similar set of measurements for counting the number of crowd in each segment, and computed texture properties like homogeneity, energy and entropy.

Having features from the segments extracted, a Gaussian Process(GP)[112] was used to regress feature vectors to the number of people per segment. The GP defines a distribution over functions, which is "pinned down" at the training points[16]. Since the classes of function that GP can model is directly dependent on the chosen kernel function, they combined the linear and the squared-exponential(RBF)(see [18, 105, 92] for more details) kernels, *i.e.*

$$k(x_p, x_q) = \alpha_1(x_p^T x_q + 1) + \alpha_2 e^{\frac{-\|x_p - x_q\|^2}{\alpha_3}} + \alpha_4 \delta(p, q) \tag{3.1}$$

The linear component of the kernel captures the dominant trend of many features which is linear(*e.g.* segment area), while the RBF component models local non-linearities that arise from a variety of factors, including occlusion, segmentation errors and pedestrian configuration (*e.g.* spacing within a segment)[16].

For this experiment, they collected an hour of video from a stationary digital camera. 2000 frames of the video were annotated as ground-truth. Moreover, a region-of-interest(ROI) was selected on the walkway(see figure 3.2), and the traveling directions (away from or towards the camera) and visible center of each pedestrian was annotated. Then the video was split into a training set, for learning the GP, and a test set for validation. The training set contains 800

frames, between frame 600 and 1399, with the remaining 1200 frames held out for testing. This dataset is available to the vision community[16].

The obtained results for crowd counting in [16] are expressed as both mean-squared-error(MSE) and mean-absolute-error(MAE) between the estimate and ground-truth. As results for crowd counting using a set of all the features, for pedestrians away from and towards the camera respectively, MSE = 4.181 and MAE = 1.621, MSE = 1.291 and MAE = 0.869 were achieved. This reasonable results given the small dataset and also its' privacy-preserving manner notwithstanding, this work, like the aforementioned methods, requires not only exhaustive data annotations and large training set, but also hand-crafting highly specialized image features that are dependent on the object class.

In order to save annotation efforts, different techniques were used to count objects. Multiple Instance Learning(MIL)[37] is a variation of supervised learning in which instances come in bags. These bags contain multiple instances. A bag is labeled positive if there is at least one example with the concept of interest, or labeled negative otherwise. The positive bag can be regarded as a set of attracting instances and the negative one as a set of repulsive instances. In large-scale Computer Vision, this approach is frequently found under the name of *weakly supervised learning*[111, 35]. There are different definitions for the term "weakly" in the literature. For instance, in [27], it is a surrogate for the concept of noisy labels such as labels provided by different supervisors with distinct quality. However, in [86], it is described for indicating imperfect annotation or even in [110] for specifying only the presence of an object in an image.

Early works used weakly supervised learning in an instantiation of the MIL framework for for inferring difficult to describe classes such as in[97] where photometric, geometric, and topological features are recognized. More recently, several works, such as[78], explore the capacity of this technique for simultaneous localization and recognition. Another work using MIL framework was count-based multiple instance learning[37]. In count-based MIL the positive bag is composed of instances where the concept appears within the range of an interval. For example, the positive bag may contain images with 5 to 10 appearances of pedestrians. A major drawback of MIL framework is that even in count-based MIL the problem is casted as a binary task and they would not be applicable in projects where the exact number of objects in the image important.

Furthermore, another approach to reduce the annotation tasks is done in [36], where the labeling process is decreased to dotting(pointing) and the counting process is addressed as image density estimation problem.

Recently, with the success of CNNs in different vision tasks, object detection systems based on deep CNN have made groundbreaking advances on several object detection problems[116, 33, 41, 49, 33] which suggests the use of this technique to learn to count objects. Several advantages can be foreseen from this application, being the most important that of learning image features from samples instead of hand-crafting highly specialized image features that are dependent on the object class[90]. Moreover, CNN have shown their capacity of knowledge transfer for a number of tasks or the ability of simultaneously performing different tasks even when trained for only one [118].

Following this line of work, Seguí et al. in [90] proposed a novel approach for counting objects' representations using deep object features. In their work, objects' features are learned by a deep counting convolutional neural network and are used to understand the underlying representation. Their proposal lies in the middle of weakly supervised learning and fully supervised learning[74]. It is similar to weakly supervised learning because the location of the concept of interest is not given. Whereas, unlike fully supervised learning in which the object boundary or bounding box

is given to the learning process, in their proposed architecture, only the multiplicity of the object is provided[90].

To this end, they defined a counting problem for even digits using *MNIST* data and demonstrated that the internal representation of the network is able to classify digits in spite of the fact that during training, no direct supervision was provided. Moreover, they present preliminary results about a deep network that is able to count the number of pedestrians in a scene[90]. Figure 3.3 illustrates their proposal at a glance in the case of representing hand-written digits:

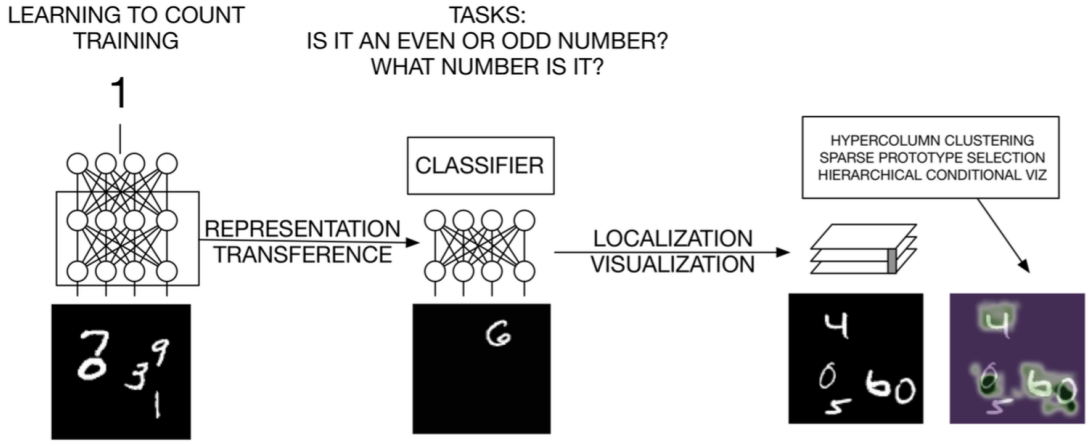

Figure 3.3: Learning to count hand-written digits problem in which the features of a CNN that has been trained to count digits can be readily used for more specific classification problems and even to localize digits in an image[90].

In [90], the main hypothesis is that the number of occurrence of objects in an image provide strong presentational information due to their possible discriminate appearance for a feature learning process to exploit. In order to verify this hypothesis, for both experiments, they considered networks of two or more convolutional layers (since CNNs instinctively handle feature learning[60]) consisting of convolutional filters, ReLU non-linearities, max-pooling layers and normalization layer, followed by one or more fully connected layers (regarding the impressive classification performance on different benchmark problems[57, 55, 24])[90].

For learning to count in the hand-written digits domain, they synthetically created a set of one million images of size 100*100 including random digits from the MNIST database with maximum 5 digit per image and with no overlapping in the images. Obtaining accuracy of 93.8% on the base network, along with results attained from training a support vector machine(SVM) with the representations learned on different layers of the network, which are incredibly promising, validates this hypothesis that counting process can be considered as a surrogate to potentially extract or infer interesting object descriptors[90].

Additionally, for learning to count the number of pedestrians in a scene, they used UCSD pedestrian database[16]. Once again, Seguí et al. 90 created set of 200.000 synthetic images each containing up to 25 pedestrians. In this experiment, the performance of the base network is MAE = 0.74 and MSE = 1.12. The results in this scenario are encouraging and reinforce the feasibility of the proposal in front of counting problems. However, still there are some deficiencies that should be obviated. For instance, hoe would a model trained by synthetic dataset perform on real dataset and in a real world problem? Or would still the model be able to learn object representations in scaled-up and more complex scenarios?

4 Methodology

This master thesis proposes an application of deep CNNs to a task of counting objects in the image. The provided system is to address all the aforestated issues which object detection and counting applications have encountered. Such as:

- Being prone to error in noisy or crowded scenes with a noticeable occlusion.
- Establishing the viability of a privacy-preserving approach.
- Painstaking hand-crafted image features which are highly dependent on the object class.
- Scrupulous data annotation for manifold data.

In addition, the deficit in the state-of-the-art [90] which would be the applicability and performance of a system trained with synthetic dataset in real world counting problem[16]. The novelty of our approach compare to the state of the art is that we hypothesize that features learned by training a counting deep CNN on a synthetic dataset, are representative enough to count the number of object of interest in a real dataset. We tackle this task as a regression problem. To the best of our knowledge, the proposed work would be the first one in which a counting system trained with synthetic images is able to be incorporated in real-world similar counting problems.

Henceforth, in the rest of this chapter, we justify our methodology along with a comparison to state-of-the-art from different aspects such as method selection, architecture, dataset and its application.

4.1 Method selection

For a long time in Computer Vision, there has been a prevailing paradigm in which we have a set of feature descriptors such SIFT [69], HOG[25], SURF[6] and many more to that can be extracted from the image with possible higher level feature building following by a classifier like Support Vector Machines (SVM) [103, 10]. In fact, for the most part, these features are not learned, but hand-crafted by some vision experts. However, they do have indeed descent performance. For instance, in one of the most successful works in object detection, in [34] the author essentially introduces a linear classifier on top of HOG features, or regarding classification approaches that work quite well, Yu et al. use all manners of features (HOG, SIFT, Color SIFT, etc) extracted from the images and consequently obtain impressive results.

The analysis of these works develop this question that in order to improve the vision systems accuracy, in which part of the system should we focus on? Should we try to enhance classifiers, should we increase the amount of data or we had better provide finer features? Parikh and Zitnick in [81] analyze the role of features by taking some of the quite successful past deformable

models[3], and replace some components of them with humans. they present identical learning tasks *i.e.* the same feature representation and the same training data, to machines and humans which allows drawing a comparison between the two. The author concludes that features are the main factor contributing to superior human performance. Furthermore, in [40], compared 39 different learning kernels with different combination features to see which kernel outperforms the rest and how it should be weighted. Although they got a big jump over the existing methods, the analysis of their results shows that the gain they obtained from the learning operators is not as dramatic as the improvement they achieved from the features itself.

Therefore, since the features are doing most of the works in these algorithms, if we improve those, we can attain better algorithms. The difficulty of feature improvement and high cost of numerous features computation on each image brought us into the application of deep learning in order to learn the features themselves rather than hand-crafting them.

In deep learning techniques, we essentially have a hierarchy of feature extractors which attempts to model high-level abstractions in data[29, 7, 8, 5, 89] where each layer of hierarchy extracts features from output of previous layer. Designing such trainable feature extractors, we would be able to build a multi-stage model which goes all the way from image pixels up to a high-level feature vector which we can feed it to a standard classifier. Thus, from the family of DL, we select deep CNNs due to its' success in may recent vision problems such as [23, 20, 109, 21], their capacity of knowledge transfer for a number of tasks and also the ability for performing different tasks simultaneously, even when it has been trained for only one task[118].

4.2 Architecture

CNNs actually have a long heritage. The origin of CNN comes from [51] where first simple cells detect local features and afterwards, complex cells pool the outputs of simple cells within a retinotopic neighborhood. Also, Fukushima in [38, 39] introduced a similar architecture with filtering layers following by pooling layers. However, the the first deep CNN architecture was designed by LeCun et al. 60 who demonstrated that this kind of architecture can perform quite well for vision tasks like digit recognition.

Following the same structure of classical CNNs, in our architecture, the image pixels are fed to a convolutional layer, where relatively small filters (windows) shift over the image (not necessarily pixel by pixel, different stride can be taken) and produce feature maps. Since convolution is a linear operation, in order to make the model non-linear, feature maps are passed to rectified linear units (ReLU) [76] which is applied to each element of the feature map independently. ReLU is currently in favor given the fact that it fastens the learning process by massively simplifying back propagation, and also avoids saturation issues(when the weighted sum is big, the output of the *tanh* or *sigmoid* activation functions saturates and the gradient tends to zero. See [47, 4] for more details.).

Thereafter, a pooling layer takes a special region of the obtained feature maps and take the maximum (or sum over the pixels of the neighborhood) pixel value as the strongest structure of that neighborhood. We chose Max-pooling since it tends to give more discrimination of the features[12]. In our model, we use *spatial* pooling. However, depending on the problem we are trying to solve, pooling might be done within feature maps[44]. The main advantage of pooling layers in the architecture can be that it results in invariance to small transformations. In addition to that, as we go up in CNNs, thanks to pooling, each of the units essentially has a larger receptive fields looking back the input so that at the top high-level layers, each unit looks

at the entire scene.

To improve the model performance, after each pooling layer in the proposed model, we used local response normalization (LRN)layer to apply a contrast normalization. This type of normalization was introduced and used for the first time in [57]. Empirically, using LRN in the architecture seems to help improving the results [52, 57]. Basically LRN introduces a local competition between features, in a way that it picks a single location in the output and it looks at a special neighborhood around that location in the input. Then it computes the local mean in that region, weight it by the Gaussian distribution, translate and make it to corresponding vectors with local mean = 0 and local standard deviation = 1(the chosen values are problem dependent. In our case, we chose $\mu = 0$ & $std = 1$). This brings out more details in the darker regions of the feature maps.

Moreover, LRN helps to scale activations at each layer better for learning by making energy surface more isotropic. That means that if we use a single learning rate for all the layers, then each gradient step tends to make much more progress[52].

And finally on top of the model, we put fully connected layers in order to do either a classification or regression strategy. In our model, since we are casting the problem as a regression problem, a *Euclidean Loss* layer is stated on top of fully connected layers to project the output as the difference between model prediction and the ground truth.

The above explanation is just to reason the selected architecture and how it can help us to overcome the deficiencies of previous related works. However, the most important fact regarding CNNs' capability to learn features is the deepness and settings of the architecture which will be attentively addressed in the next chapter.

4.3 Datasets

Here, starting with a short review of data for CNN in vision, we express a succinct reasoning about the three datasets we used for two distinct but somehow related experiments. Later on, in the implementation section, the specifications and preprocessing phases regarding each dataset will be described in details.

In spite of remarkable performance of CNN in some simple recognition benchmarks [23, 20, 109, 21], until recently the models' performances were poor at more complex datasets [45] due to lack of labeled input samples to train the network parameters with. It was with the creation of ImageNet[28] and GPU implementation[57] (50x speedup over CPU) that efficiency of deep CNN in vision tasks became crystal clear.

4.3.1 MNIST pool of digits dataset

For the first empirical experience in our work, we created a large set of images each containing up to 15 MNIST digits. Images are automatically labeled with the number of even digits in each image. Using this dataset, we can examine the digits' representations learned by our deep CNN are not dependent on the specific task we are dealing with. It is done by using our model's learned features for other tasks and analyze the results.

4.3.2 Synthetic crowd counting dataset

For the second task, since we needed a large dataset to train all the parameters of our model, we synthetically created a large dataset of pedestrians in a walkway with maximum 25 people in

each image. Achieving success on crowd counting system using CNN on our synthetic dataset, we save enormous amount of time for data annotation and feature detection.

4.3.3 UCSD crowd counting dataset

However, we need our model to perform well in practice rather than in theoretical experiments. Therefore, we used UCSD crowd counting dataset [16] to validate the performance of our model on it and also make a comparison with their system in [16] to see if we can obtain better results while decreasing feature extraction and labeling efforts.

5 Implementation

In this chapter, we provide a detailed implementation of our proposed methodology. We start with presenting the platform we incorporated to shape and design our model. Then we demonstrate our network's architecture in detail. Finally, we attempt to give insight into the datasets we used to train and test our model.

5.1 Caffe deep learning platform

Caffe is a clean and modifiable framework for state-of-the-art deep learning algorithms and a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB bindings for training and deploying general-purpose convolutional neural networks and other deep models efficiently on commodity architectures. It powers on-going research projects and large-scale industrial applications in vision, speech and multimedia by CUDA GPU computation (CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA[1]), processing over 40 million images a day on a single K40 or Titan GPU[54]. The main components of Caffe architecture are listed and succinctly explained in below:

1. **Data storage:** Caffe stores and communicates data in 4-dimensional arrays called *blobs*. Blobs provide a unified memory interface, holding batches of data, parameters, or parameter updates. Blobs conceal the computational overhead by synchronizing from the CPU host to the GPU device as needed. Caffe supports some data sources such as LevelDB or LMDB, HDF5, MemoryData, ImageData, etc. However, large-scale data is stored in LevelDB data bases since it reads the data directly from memory[2].

2. **Layers:** A caffe layer takes blobs as input and yields one or more as output. In a network, each layer plays two important roles: a forward pass that takes the inputs and produces the outputs, and a backward pass that takes the gradient with respect to the output, and computes the gradients with respect to the parameters and to the inputs, which are in turn back-propagated to earlier layers [54].

   Caffe affords a exhaustive set of layers including: convolution, pooling, fully connected, nonlinearities like rectified linear and logistic, local response normalization, element-wise operations, and losses like softmax and hinge [54].

3. **Networks and run mode:** Caffe ensures the correctness of the forward and backward passes for any directed acyclic graph of layers. A typical network begins with a data layer laying at the bottom going up to the loss layer that computes tasks' objectives. The network is run on CPU or GPU independent of the model definition.

4. **Training a network:** Training phase in Caffe is done by classical stochastic gradient descent algorithm. When training, images and labels pass through different layers to be fed to the final prediction into a classification layer that produces the loss and gradients which train the whole network. Figure 5.1 illustrates a typical example of a Caffe network.

   Finetuning, the adaptation of an existing model to new architectures or data, is a standard method in Caffe. Caffe finetunes the old model weights for the new task and initializes new weights as needed. This capability is essential for tasks such as knowledge transfer [30], object detection [41], and object retrieval [46] [54].

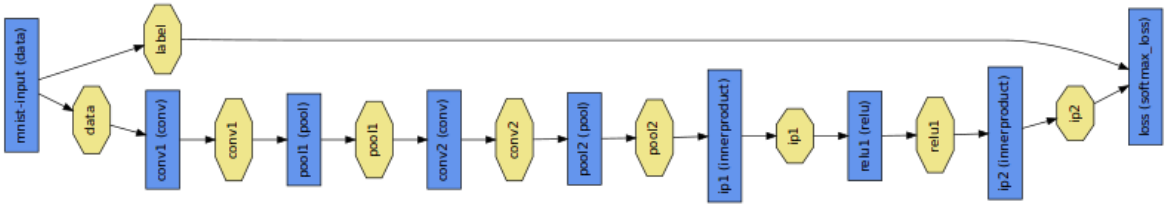

Figure 5.1: An MNIST digit classification example of a Caffe network, where blue boxes represent layers and yellow octagons represent data blobs produced by or fed into the layers[54].

We decided to use Caffe because, it addresses computation efficiency problems (as likely the fastest available implementation of deep learning frameworks), adheres to software engineering best practices, providing unit tests for correctness and experimental rigor and speed for deployment. It is also well-suited for research use,due to the careful modularity of the code, and the clean separation of network definition (usually the novel part of deep learning research) from actual implementation[54]. Moreover, providing Python wrapper which exposes the solver module for easy prototyping of new training procedures.

5.2 The architecture

In learning features or object representations for vision tasks and by the use of neural networks, the depth of network plays an important role. The deeper the model, the better it learns. However, issues like overfitting and underfitting should not be left neglected. Having Caffe platform introduced, we propose the designed network for two experiments we did regarding learning to count problems. Therefore, in this section, networks' settings and architectures for even digit recognition and crowd counting problems will be described separately.

5.2.1 Even digit recognition

For learning to count even digits problem, since we used MNIST dataset to generate our dataset, we decided to start with an architecture similar to the classic MNIST hand-written digit recognition problem[63]. From there, we modified the architecture to optimize the performance of the network.

In our network, the data layer fetches the images and labels from the disk, passes it through, the first convolutional layer with 20 filters, each of size 15*15 followed by a ReLU non-linearity and LRN normalization layer. Then the output is pooled by the size of 2*2. This process repeats again but this time with the second convolutional layer having 50 filters of size 3*3. In all convolution and pooling layers, the $stride = 1$ and $padding = 1$ are considered. Afterwards,

the output of the second pooling layer is fed to two fully connected (inner product) layers with respectively 64 and 1 number of outputs (since the problem is approached as a regression task). Both fully connected layers are followed by ReLU non-linearities. Figure 5.3 shows a schematic of the architecture. In addition, parameters of the network are set as below:

- **Learning rate:** The basic learning rate is 0.0001. However, for our experiment we chose *multi-step* learning policy in which, after each *stepsize*=40000 iterations, the learning rate drops by the rate of $\gamma = 0.1$. This initialization is based on rules of thumbs used in [57].
- **Momentum:** We use momentum $\mu = 0.9$. This selection also is based one rules of thumbs. Because, momentum setting μ effectively multiplies the size of our updates by a factor of $\frac{1}{1-\mu}$. Hence, changes in momentum and learning rate ought to be accompanied with an inverse correlation. When momentum $\mu = 0.9$, we have an effective update size of 10 since we also drop the learning rate by the factor of $\gamma = 0.1$.
- **Weight decay:** Weight decay as a penalty term to the error function, has a constant value of 0.0005. This decay constant is multiplied to the sum of squared weights.

We should also mention that at the top layer of the network, we used *Euclidean Loss* layer to compute the euclidean distance between the predictions and ground truth.

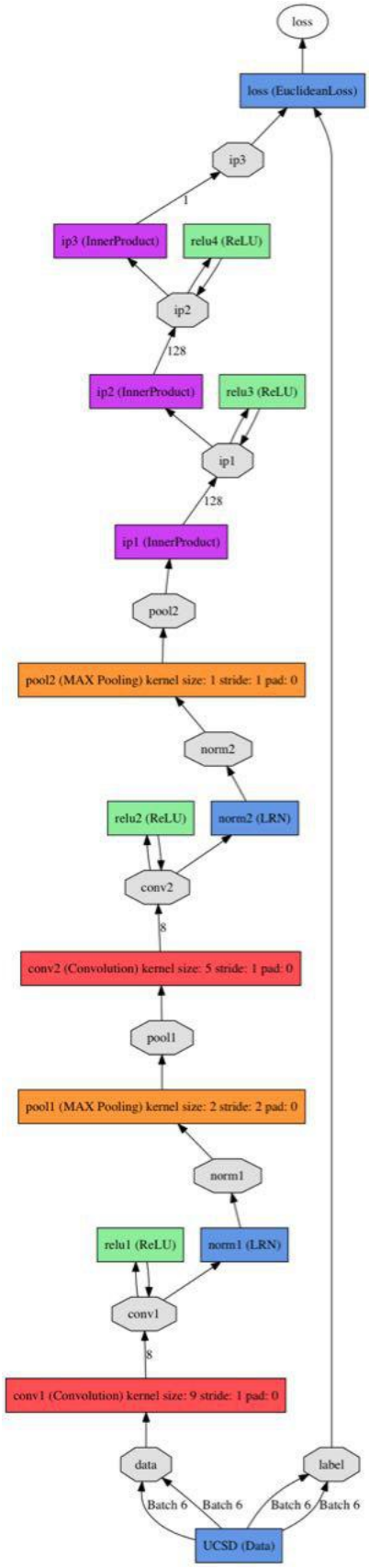

Figure 5.2: Proposed network architecture for Even digits recognition task

5.2.2 Crowd counting

In the case of counting pedestrians task, we applied the same settings to a different architecture. This time, due to more complexity of images, we considered a deeper network. Here the data blobs pass through 4 convolutional layers. First convolutional layer has 4 filters, each with 5*5 kernel and the other 3 layers have again 4 filters but each of size 3*3. Similar to the previous model, each convolutional layer is followed by ReLU nonlinearity layer and LRN normalization layer. Also the stride and padding values for all the convolutional layers are respectively equal to 1 and 0.

In order to not lose information, we used merely two pooling layers for the first two convolutional layers. Each pooling layer has a kernel size of 2*2 with stride = 1 and padding = 0.

There are three fully connected layers to regress the number of pedestrians in images. The first two fully connected layers have 16 outputs each and are connected to ReLU nonlinearity layers. The last layer however, with solely one output, passes the models' prediction of the number of pedestrians to the Euclidean loss layer to calculate the sum of squares of differences of its two inputs, the true labels and predictions.

To the best of our knowledge and experience, the designed architectures outperform the other architectures while fasten the training phase. However, apart from the basic knowledge about network architectures, hyper-parameters initialization and some rules of thumbs of successful experiences in similar works, the rest of design has been done intuitively.

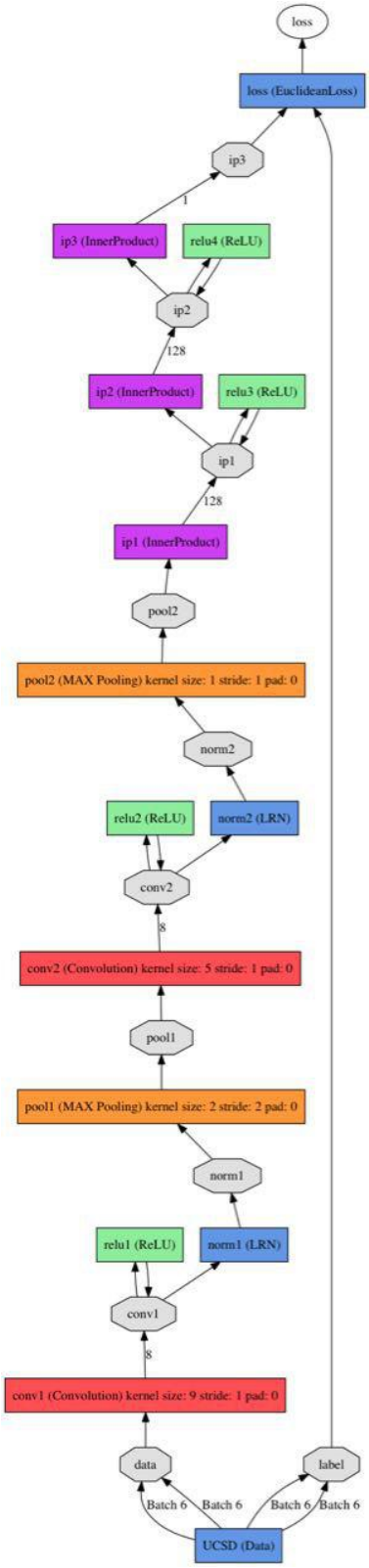

Figure 5.3: Proposed network architecture for Even digits recognition task

5.3 The datasets

Now, we delve into the data processing part of this work by introducing three different datasets we generated or chose for our empirical experiments. To that end, we provide a detailed explanation of the approaches and methods used to generate and improve each dataset.

5.3.1 Even-odd digits dataset

For the first analysis,we used original MNIST dataset to create our set of images[61]. MNIST dataset contains a training set of 60,000 examples and a test set of 10,000 examples. Each image has a size of 28*28 with one random hand-written digit centered in the image. An example of original MNIST is depicted in the below figure.

Figure 5.4: An example of original MNIST data with hand-written digit number 4 in the image.

Our Even-odd handwritten dataset contains images of size 100*100. Each image is filled with 0 up to 15 randomly selected digits from MNIST dataset. Digits are resized to 18*18 pixels and randomly put in the image. The images are created with controlled overlapping by ensuring that two different numbers are 18 pixels away from each other, i.e. the distance between two digits centers is larger than 18 pixels. For the training process, images are labeled with the number of even digits present in each image. Figure 5.5 illustrates some examples of even-odd digits dataset with different number of even digits in images.

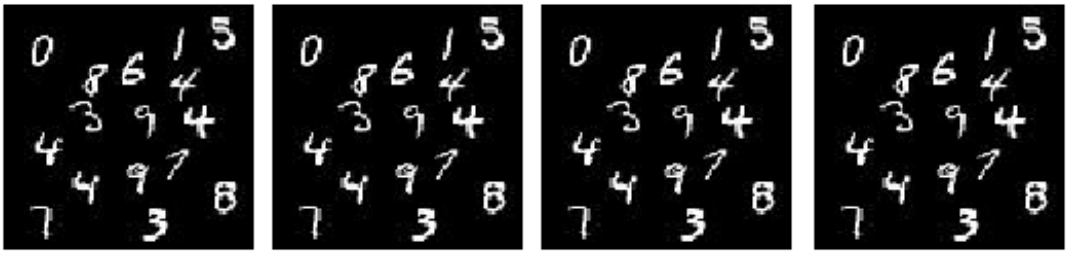

Figure 5.5: An example of even-odd digits images. Form left to right, images contain 0, 5, 10 and 15 even digits.

This dataset has in total 1 million images, 800,000 images for training set and 200,000 as the test. Also, the dataset is uniformly generated, meaning that for instance, the number of images containing 0 even digits are equal to the number of images containing 15 even digits.

5.3.2 Synthetic pedestrian dataset

5.3.3 UCSD crowd counting dataset

References

[1] (12008). Cuda.

[2] (2008).

[3] Albrecht, T., Luthi, M., and Vetter, T. (2015). Deformable models. *Encyclopedia of Biometrics*, pages 337–343.

[4] Amit, D. J., Gutfreund, H., and Sompolinsky, H. (1987). Statistical mechanics of neural networks near saturation. *Annals of physics*, pages 30–67.

[5] Arel, I., Rose, D. C., and Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, pages 13–18.

[6] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer.

[7] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[8] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 1798–1828.

[9] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

[10] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

[11] Bottou, L. (2010). Large-scale machine learning with stochasticgradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[12] Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

[13] Brandtberg, T. and Walter, F. (1998). Automated delineation of individual tree crowns in high spatial resolution aerial images by multiple-scale analysis. *Machine Vision and Applications*, 11(2):64–73.

[14] Brostow, G. J. and Cipolla, R. (2006). Unsupervised bayesian detection of independent motion in crowds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 594–601. IEEE.

[15] Chan, A. and Vasconcelos, N. (2013). Ground truth annotations for ucsd dataset. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Chan, A. B., Liang, Z.-S. J., and Vasconcelos, N. (2008). Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7. IEEE.

[17] Chan, A. B. and Vasconcelos, N. (2008). Modeling, clustering, and segmenting video with mixtures of dynamic textures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5):909–926.

[18] Chang, Y.-W., Hsieh, C.-J., Chang, K.-W., Ringgaard, M., and Lin, C.-J. (2010). Training and testing low-degree polynomial data mappings via linear svm. *The Journal of Machine Learning Research*, 11:1471–1490.

[19] Cho, S.-Y., Chow, T. W., and Leung, C.-T. (1999). A neural-based crowd estimation by hybrid global learning algorithm. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(4):535–541.

[20] Ciresan, D. and Meier, U. (2015). Multi-column deep neural networks for offline handwritten chinese character classification. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–6. IEEE.

[21] Cireşan, D., Meier, U., Masci, J., and Schmidhuber (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, pages 333–338.

[22] Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE.

[23] Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1135–1139. IEEE.

[24] Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, page 1237.

[25] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

[26] Davies, A. C., Yin, J. H., and Velastin, S. A. (1995). Crowd monitoring using image processing. *Electronics & Communication Engineering Journal*, 7(1):37–47.

[27] Dekel, O. and Shamir, O. (2009). Good learners for evil teachers. In *Proceedings of the 26th annual international conference on machine learning*, pages 233–240. ACM.

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

[29] Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, pages 197–387.

[30] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*.

[31] Dong, L., Parameswaran, V., Ramesh, V., and Zoghlami, I. (2007). Fast crowd segmentation using shape indexing. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[32] Doretto, G., Chiuso, A., Wu, Y. N., and Soatto, S. (2003). Dynamic textures. *International Journal of Computer Vision*, 51:91–109.

[33] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154.

[34] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32:1627–1645.

[35] Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, pages II–264. IEEE.

[36] Flaccavento, G., Lempitsky, V., Pope, I., Barber, P., Zisserman, A., Noble, J., and Vojnovic, B. (2011). Learning to count cells: applications to lens-free imaging of large fields. *Microscopic Image Analysis with Applications in Biology*, 1:3.

[37] Foulds, J. and Frank, E. (2010). A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25:1–25.

[38] Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, pages 121–136.

[39] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, pages 193–202.

[40] Gehler, P. and Nowozin, S. (2009). On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 22–228. IEEE.

[41] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[42] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323.

[43] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

[44] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.

[45] Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. California Institute of Technology.

[46] Guadarrama, S., Rodner, E., Saenko, K., Zhang, N., Farrell, R., Donahue, J., and Darrell, T. (2014). Open-vocabulary object retrieval. In *Robotics: science and systems*, volume 2, page 6.

[47] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 993–1001.

[48] Haralick, R. M., Shanmugam, K., and Dinstein, I. H. (1973). Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, pages 610–621.

[49] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(9):1904–1916.

[50] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[51] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, pages 106–154.

[52] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE.

[53] Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231.

[54] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM.

[55] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[56] Kong, D., Gray, D., and Tao, H. (2005). Counting pedestrians in crowds using viewpoint invariant training. In *BMVC*. Citeseer.

[57] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[58] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[59] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[60] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989a). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[61] LeCun, Y., Cortes, C., and Burges, C. J. (1998). The mnist database of handwritten digits.

[62] LeCun, Y. et al. (1989b). Generalization and network design strategies. *Connections in Perspective. North-Holland, Amsterdam*, pages 143–55.

[63] LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., et al. (1995). Comparison of learning algorithms for handwritten digit recognition. In *Comparison of learning algorithms for handwritten digit recognition*.

[64] LeCun, Y., Kavukcuoglu, K., Farabet, C., et al. (2010). Convolutional networks and applications in vision. In *ISCAS*, pages 253–256.

[65] Leibe, B., Schindler, K., and Van Gool, L. (2007). Coupled detection and trajectory estimation for multi-object tracking. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[66] Leibe, B., Seemann, E., and Schiele, B. (2005). Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 878–885. IEEE.

[67] Lempitsky, V. and Zisserman, A. (2010). learn. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 1324–1332. Curran Associates, Inc.

[68] Lin, S.-F., Chen, J.-Y., and Chao, H.-X. (2001). Estimation of number of people in crowded scenes using perspective transformation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(6):645–654.

[69] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

[70] Mahadevan, V., Li, W., Bhalodia, V., and Vasconcelos, N. (2010). Anomaly detection in crowded scenes. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, 2010*.

[71] Marana, A., Costa, L. d. F., Lotufo, R., and Velastin, S. (1998). On the efficacy of texture analysis for crowd monitoring. In *Computer Graphics, Image Processing, and Vision, 1998. Proceedings. SIBGRAPI'98. International Symposium on*, pages 354–361. IEEE.

[72] Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

[73] Mitchell, T. M. et al. (1997). Machine learning.

[74] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

[75] Moody, J., Hanson, S., Krogh, A., and Hertz, J. A. (1995). A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4:950–957.

[76] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

[77] Ng, A. and colleagues (2013). http://ufldl.stanford.edu/tutorial/supervised/ optimization-stochasticgradientdescent/.

[78] Nguyen, M. H., Torresani, L., de la Torre, F., and Rother, C. (2009). Weakly supervised discriminative localization and classification: a joint learning process. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1925–1932. IEEE.

[79] Oliva, A., Torralba, A., Castelhano, M. S., and Henderson, J. M. (2003). Top-down control of visual attention in object detection. In *Image processing, 2003. icip 2003. proceedings. 2003 international conference on*, volume 1, pages I–253. IEEE.

[80] Paragios, N. and Ramesh, V. (2001). A mrf-based approach for real-time subway monitoring. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–1034. IEEE.

[81] Parikh, D. and Zitnick, C. L. (2010). The role of features, algorithms and data in visual recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2328–2335. IEEE.

[82] Pollock, R. J. (1996). *The automatic recognition of individual trees in aerial images of forests based on a synthetic tree crown image model*. PhD thesis, Concordia University.

[83] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

[84] Rabaud, V. and Belongie, S. (2006). Counting crowded moving objects. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 705–711. IEEE.

[85] Rahmalan, H., Nixon, M. S., and Carter, J. N. (2006). On crowd density estimation for surveillance. In *Crime and Security, 2006. The Institution of Engineering and Technology Conference on*, pages 540–545. IET.

[86] Raykar, V. C., Yu, S., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., and Moy, L. (2009). Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings of the 26th annual international conference on machine learning*, pages 889–896. ACM.

[87] Regazzoni, C. S. and Tesei, A. (1996). Distributed data fusion for real-time crowding estimation. *Signal Processing*, 53(1):47–63.

[88] Schmidhuber, J. (2015a). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[89] Schmidhuber, J. (2015b). Deep learning in neural networks: An overview. *Neural Networks*, pages 85–117.

[90] Seguí, S., Pujol, O., and Vitria, J. (2015). Learning to count with deep object features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 90–96.

[91] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

[92] Shashua, A. (2009). Introduction to machine learning: Class notes 67577. *arXiv preprint arXiv:0904.3664*.

[93] Song, H. A. and Lee, S.-Y. (2013). Hierarchical representation using nmf. In *Neural Information Processing*, pages 466–473. Springer.

[94] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147.

[95] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[96] Taylor, G. W., Fergus, R., LeCun, Y., and Bregler, C. (2010). Convolutional learning of spatio-temporal features. In *Computer Vision–ECCV 2010*, pages 140–153. Springer.

[97] Todorovic, S. and Ahuja, N. (2006). Extracting subimages of an unknown category from a set of images. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 927–934. IEEE.

[98] Tutorial, D. L. (2014). Lisa lab. *University of Montreal*.

[99] Umbaugh, S. E. (1997). *Computer Vision and Image Processing: A Practical Approach Using Cviptools with Cdrom*. Prentice Hall PTR.

[100] University, S. (2013). http://white.stanford.edu/teach/index.php/an_introduction_to_convolutional_neural_networks.

[101] Vaidya, J., Clifton, C. W., and Zhu, Y. M. (2006). *Privacy preserving data mining*, volume 19. Springer Science & Business Media.

[102] Valera, M. and Velastin, S. A. (2005). Intelligent distributed surveillance systems: a review. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 152, pages 192–204. IET.

[103] Vapnik, V. and Chervonenkis, A. (1964). A note on one class of perceptrons. *Automation and remote control*, 25(1).

[104] Vapnik, V. N. and Vapnik, V. (1998). *Statistical learning theory*, volume 1. Wiley New York.

[105] Vert, J.-P., Tsuda, K., and Schölkopf, B. (2004). A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70.

[106] Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y. (2004). State-of-the-art in privacy preserving data mining. *ACM Sigmod Record*, 33(1):50–57.

[107] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.

[108] Viola, P., Jones, M. J., and Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161.

[109] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066.

[110] Wang, S., Joo, J., Wang, Y., and Zhu, S.-C. (2013). Weakly supervised learning for attribute localization in outdoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3111–3118.

[111] Weber, M., Welling, M., and Perona, P. (2000). *Unsupervised learning of models for recognition*. Springer.

[112] Williams, C. K. and Rasmussen, C. E. (2006). Gaussian processes for machine learning. *the MIT Press*, 2(3):4.

[113] Williams, D. R. G. H. R. and Hinton, G. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[114] Wu, B. and Nevatia, R. (2005). Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 90–97. IEEE.

[115] Yu, Y., Zhang, J., Huang, Y., Zheng, S., Ren, W., Wang, C., Huang, K., and Tan, T. (2010). Object detection by context and boosted hog-lbp. In *VOC Workshop Talk*, page 104.

[116] Zhang, Y., Sohn, K., Villegas, R., Pan, G., and Lee, H. (2015). Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 249–258.

[117] Zhao, T. and Nevatia, R. (2003). Bayesian human segmentation in crowded situations. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–459. IEEE.

[118] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495.