Deep Learning (2IMM10) - Introduction

Vlado Menkovski

Spring 2020

1 Introduction

Learning outcomes of this chapter:

- Understand when to use Machine Learning
- Be able to motivate when to use Deep Learning
- Understand and be able to communicate what are the advantages of learning representations with neural network models

1.1 Motivation - ML

Many of the systems we aim to develop have the goal of enabling automation and in turn efficiency. Examples of such developments exist in domains such as computer vision, speech recognition, and natural text understanding. Building such solutions can be done by carefully designing a set of rules that guarantee the outcome will satisfy the given goal. To be able to do this we need to understand the domain sufficiently well. In many cases, however, such solutions can become increasingly complex.

One example would be the launch of a satellite in orbit. The mechanics involved in achieving orbit and placing the satellite in a particular location and velocity require highly complex calculations over sensory information. The control of the vehicles is also very complex and requires a large set of rules to achieve its desired operation. However complex, this problem does not necessitate a learning algorithm because there is sufficient understanding that allows for a successful solution that meets the desired goal. Any algorithm that would rather learn from observation would not exceed the performance of an expertly designed solution.

So, why are many problems that are much easier for humans so much harder to solve algorithmically using expert knowledge — instead requiring learning from examplesfig. 1?

Let us take two such problems as examples: driving a car and translating a text from one language to another. What are the unique properties of these two problems that make them well-suited for Machine Learning approaches?

It basically boils down to our inability to express the goals of these task with rules sufficiently precisely. This can either stem from our lack of understanding or from the underlying complexities in the system. In many such cases the concepts that we use to describe the system do not fully explain the state and evolution of it.

Many such examples can be found in biological systems. For example we can measure the genetic

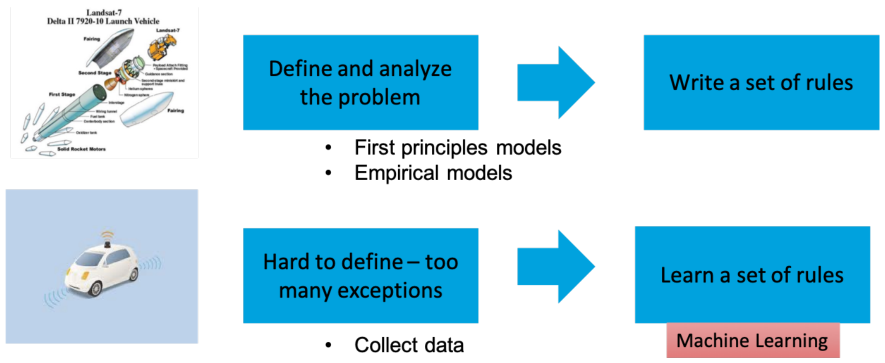

Define and analyze the problem
Write a set of rules
• First principles models
• Empirical models
Hard to define – too many exceptions
Learn a set of rules
• Collect data
Machine Learning

Figure 1: Traditional modeling versus machine learning.

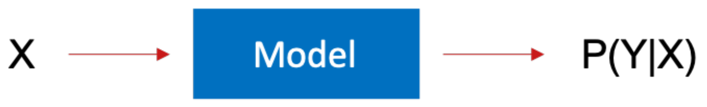
code of a cell, we can also measure many other molecules present in it. However, these measurements typically destroy the cell so we cannot measure the evolution of the values in time precisely enough to develop a sufficiently accurate model of this system.

In other cases the sheer number of rules is so large that we cannot hope to express them fully. Imagine a scenario of detecting cats and dogs in images by looking at pixel values. Expressing all the ways in which combinations of pixel values determine whether a picture contains a cat or a dog, in rules is a very challenging task.

Nevertheless, even without full understanding from observing the system we may be able to make predictions. For example, we may learn that when a gene in the cell is expressed a particular behaviour can be expected, or the statistics of the pixel value of an image may be indicative of whether we are looking at airoplaes or zebras.

Such examples, motivate the use of Machine Learning (ML). ML gives us tools to build models from examples (observations) rather than data.

Question: What are the challenges self driving cars and translation pose that motivate ML?



$$f: X \rightarrow Y$$

$$f: X \rightarrow P(Y|X)$$

f - family of functions parameterized by θ

Training = find the best θ

Figure 2: Formulation of a machine learning model as a parameterized function.

Motivation - ML - Definition

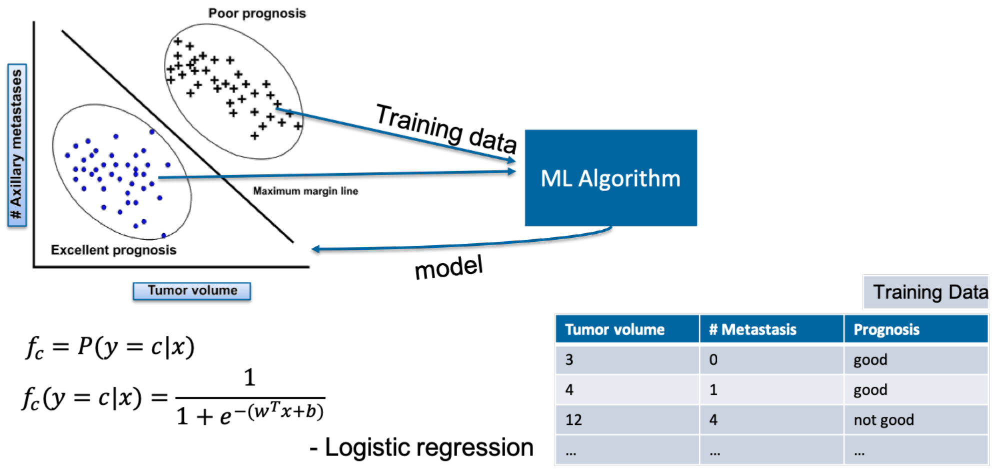
A Machine Learning model can be formulated as a function that maps its input X to an output Y and is parameterized by a parameters θ. In this formulation we assume that there is a set of rules that can determine the value of Y fully from the value of X.

Note: We typically use probability theory to express uncertainties about such maps. A ML model would express the probability of Y receiving certain values given a value for X.

ML algorithms are then developed to find good parameters θ. Typically this is done through an optimization process. If the model maps some input X to some output $\hat{Y}$, and the correct values would have been Y, some error, or loss, between $\hat{Y}$ and Y is computed. The algorithm tries to minimize this error by adjusting the values of θ.

Motivation - ML - Limitations

These algorithms have their own limitations. When the number of input variables is small, and there is a strong correlation between the variables and the correct outcomes, ML algorithms tend to perform well. However, as dimension of the input grows, and the output becomes less dependent on specific variables, ML algorithms often have a hard time finding good parameters for the model. For example, for large images the value of a single pixel is only very weakly related to whether the image contains a cat or a dog. Consequently ML algorithms have much more difficulty classifying large images than small images.



$$f_c = P(y = c|x)$$

$$f_c(y = c|x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

Tumor volume	# Metastasis	Prognosis
3	0	good
4	1	good
12	4	not good
…	…	…

Figure 3: Data set for the prognosis of cancer patients. Blue circles indicate an 'Excellent' prognosis and black plus-signs indicate a 'poor' prognosis.

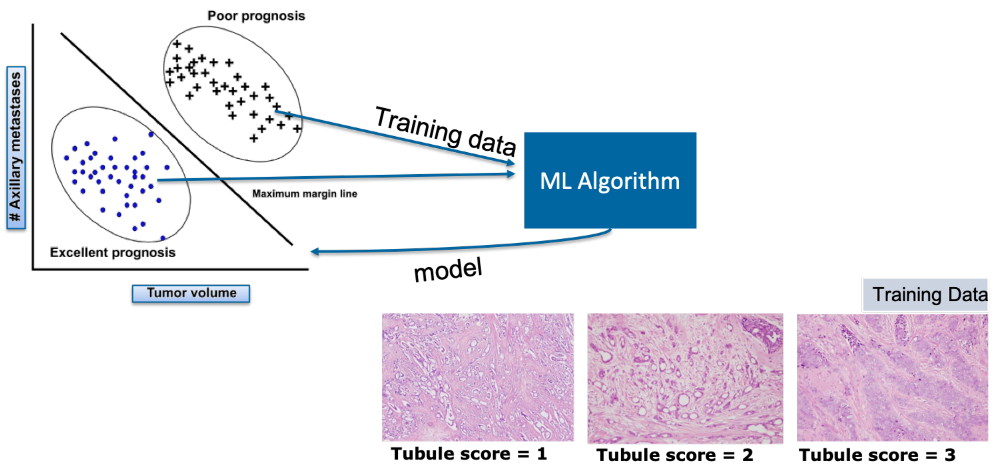


Figure 4: The data sets belonging to the two different examples.

Motivation - ML - Example

Let's look at a fictional dataset of cancer patients. Figure 3 illustrates a dataset with two features:

- Tumor volume
- Number of auxiliary metastasis

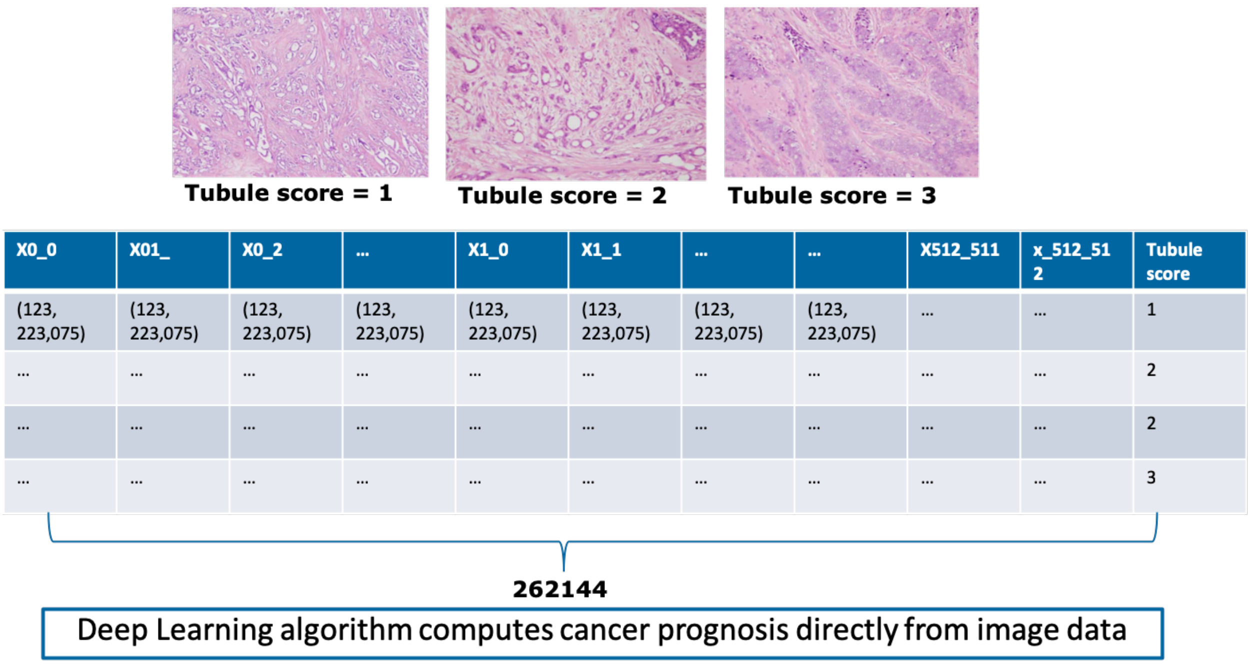
The target variable for this dataset is the prognosis for the patient. This is a discrete variable that can have two values:

- Excellent
- Poor

In the figure each datapoint is labelled with blue circle for 'Excellent' and black plus sign for 'Poor'. This dataset has a strong correlation between the two features and the target variable. Specifically, the datapoints with different labels form clear clusters that can be linearly separated, i.e. we can draw a straight line between the data points with different labels. In other words, we can define a linear combination of the two features that determine the value of the target variable.

The parameters of that linear combination (that is our model) are what the ML algorithms need to determine from the available data.

Question: Why can't we specify rules from which we can determine the prognosis given the values of the features for this problem?


Tubule score = 1
Tubule score = 2
Tubule score = 3
262144
Deep Learning algorithm computes cancer prognosis directly from image data

X0_0	X01_	X0_2	...	X1_0	X1_1	...	...	X512_511	x_512_512	Tubule score
(123, 223,075)	(123, 223,075)	(123, 223,075)	(123, 223,075)	(123, 223,075)	(123, 223,075)	(123, 223,075)	(123, 223,075)	...	...	1
...	...	...	...	...	...	...	...	...	...	2
...	...	...	...	...	...	...	...	...	...	2
...	...	...	...	...	...	...	...	...	...	3

Figure 5: The input to our model consists of the values for all pixels. That makes 262144 three-dimensional variables.

Motivation - ML - Example

In another setting, we are facing a similar task of determining the prognosis of cancer patients. However, in this case the data has significantly different properties. The dataset in this case consists of optical images of biopsy samples. The images of the samples show different structures of the tissue that reflect the severity of the disease. Specifically, the samples are from breast tissue and our aim is to determine the prognosis of breast cancer. In the bottom row of Figure 4 you can see three different images. Each of the images shows tissue with a different 'tubule score' which will be our target variable. The 'tubular score' is part of the diagnostic process and involved in determining the prognosis.

In contrast to the previous example where there were two input variables that were strongly correlated with the target variable, now each of the pixels in the image is a variable that we need to consider. As illustrated in Figure 5, given in a table format we can observe 262144 variables.

It is no surprise that the values of each pixel are very weakly correlated with our target, the tubule score. Therefore it is evident that training a model that maps each parameter value to the target value would be exceedingly difficult.

As our main challenge is that these features are far removed from the target variable, can we do something to improve this? In other words, can we engineer features that would be more useful for predicting the target variable?

Motivation - ML - Feature Engineering

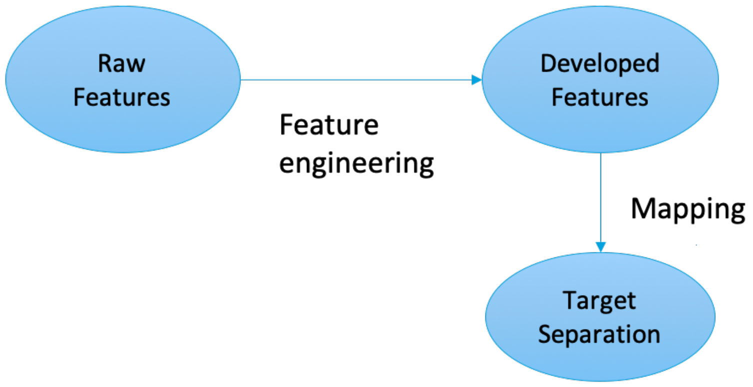


Figure 6: Feature engineering

One approach to address the challenge of using 'low-level' or 'raw' features for building ML models is to refine them in some way or to build more informative features, or 'high-level' features, from them. This approach is referred to as *Feature Engineering*.

The goal of feature engineering is to develop features that are more informative and for which a ML algorithm can develop a sufficiently good model. The assumption here is that we can use some kind of knowledge or expertise that will allow us to develop such 'high-level' features from the 'low-level' features. This is in a way analogous to the expertly designed rules, however now we are designing rules to develop features. The next step is to use a ML algorithm to develop a model using those features (fig. 6). In contrast to the expert systems the ML algorithms can actually indicate the quality of the developed features. If a feature has a weak relation to the target value, it will end up not being used in the model or the model will have very low sensitivity to that feature. So, in principle we can use ML tools to improve the feature engineering process (fig. 7).

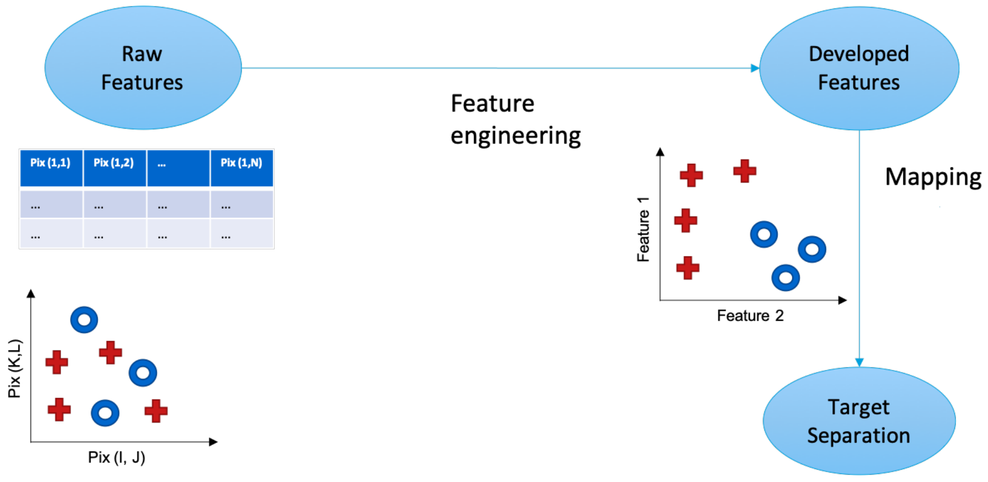
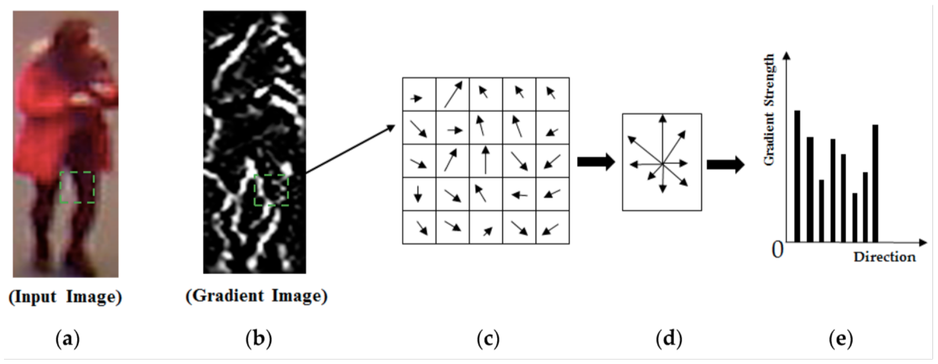


Figure 7: A schematic overview of feature engineering.

Question: What would be a possibility for a feature engineering target given our tubule score example? Do you see any patterns in the images for which we can develop feature engineering algorithms?



Nguyen, Dat Tien, et al. "Person recognition system based on a combination of body images from visible light and thermal cameras." *Sensors* 17.3 (2017): 605.

Figure 8: An illustration of a Histogram of Gradients

One problem with feature engineering is that in many cases the 'low-level' features are very high dimensional[1] and domain knowledge is not available (or at least not to a level that would allow for developing hard-coded rules).

One example of feature engineering using in general purpose image analysis such as the histogram of oriented gradients (HOG) fig. 8. This technique counts the occurrences of gradient orientations in small patches of the image. These are technique is very useful in detecting objects from background and is helpful in describing some objects, but falls short of capturing the high level concepts that we aim for in analysing images of various downstream tasks. Therefore, it is usually the case that we need to custom engineer features for specific tasks.

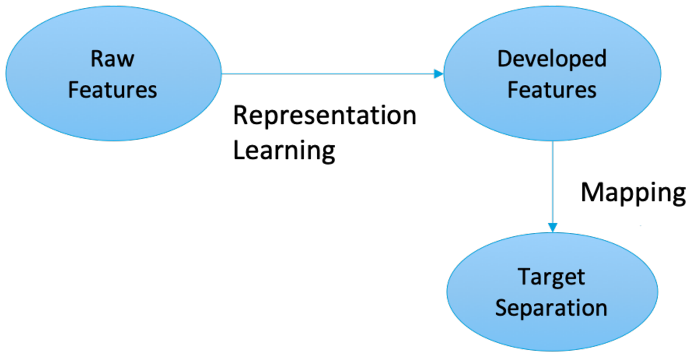


Figure 9: A schematic overview of representation learning

1.2 Motivation - ML - Representation Learning

As mentioned earlier, when feature engineering, we often face similar challenges as when trying to develop rule-based models. High quality features are difficult to design, and the process takes time. Moreover, features developed for specific tasks are not necessarily useful for other tasks. Finally,

[1]For example, with the pictures of biopsy samples we have 512 by 512 pixels all taking 3 values. That makes our input $512 \cdot 512 \cdot 3 = 786432$ dimensional.

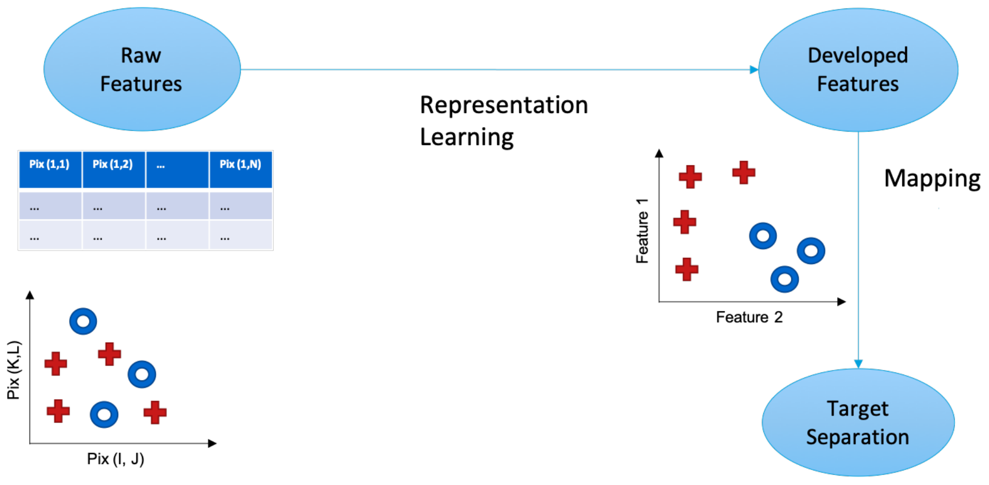


Figure 10: Representation learning end-to-end

in many cases (e.g. speech recognition) feature engineering has never managed to deliver features that were informative enough to enable the building of models with sufficient level of accuracy.

This invites the question of whether we can use ML to learn features in the same way that we learn models (or learn the parameters of models). Approaches trying to do this are referred to as *Representation Learning* (fig. 9). Much of this course revolves around representation learning, specifically learning representations of high-dimensional data for different tasks.

The setting is is the following: in many problems we face high-dimensional (low-level) data for which individual variables are far removed from a target variable and hence for which finding a map from the features to the target is difficult.

We study how to use artificial neural networks (ANN) to efficiently learn such maps. ANN models consists of a sequence of 'layers' that apply non-linear transformations to their inputs. This structure allows for learning hierarchical features with increasing level of complexity. This way, the neural network model can compose features of features of features in a hierarchical way such that sufficiently informative high-level features can then have a simple (even linear) map to the target variable (fig. 10).

This type of representation learning can also be referred to as *end-to-end representation learning* as the loss function[2] from the end of the model is used to learn the features at all levels back to the beginning (fig. 11). Moreover, as these ANN models can be deep (consisting of many layers) for certain types of data and tasks, the family of methods using them is referred to as *Deep Learning*.

1.3 Course overview

In this course we study the following topics (fig. 12 [3]):

[2]That is the function we try to minimize in order to train the model. Often this function is a measure of the error between predictions and correct outcomes.

[3]orange boxes

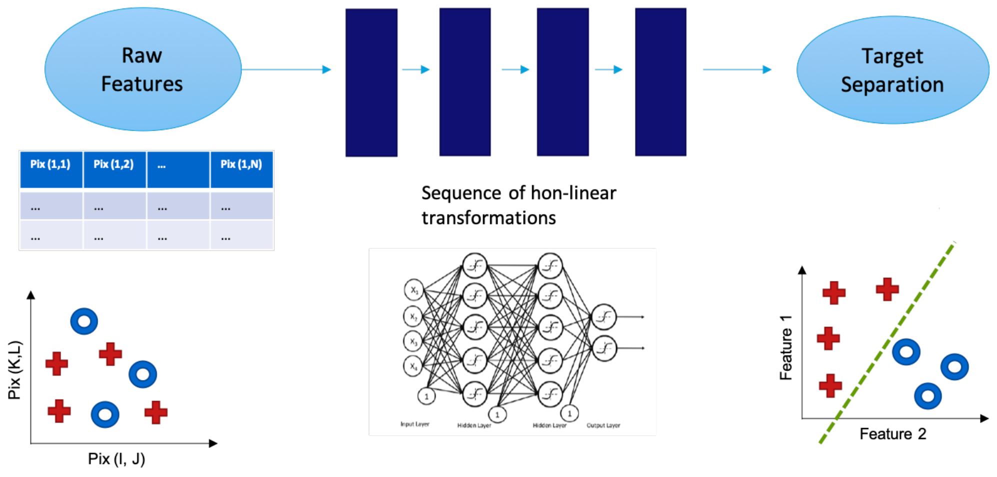


Figure 11: Representation learning with a neural network

- Artificial Neuron
- Stochastic gradient descent and backpropagation
- Multilayered perceptron (MLP)
- Convolutional Neural Networks (CNN)
- Recurrent Neural Networks (RNN)
- (Deep) metric learning
- Generative models
  - Variational Autoencoders (VAE)
  - Autoregressive models
  - Generative Adversarial Networks

In addition to these topics that follow techniques for building models, in the course we also cover solutions for downstream tasks using these techniques (fig. 12[4]). These include:

- Word embeddings
- Models for spatially distributed data
- Models for sequential data
- One shot learning
- Density estimation
- Image generation

[4]blue boxes

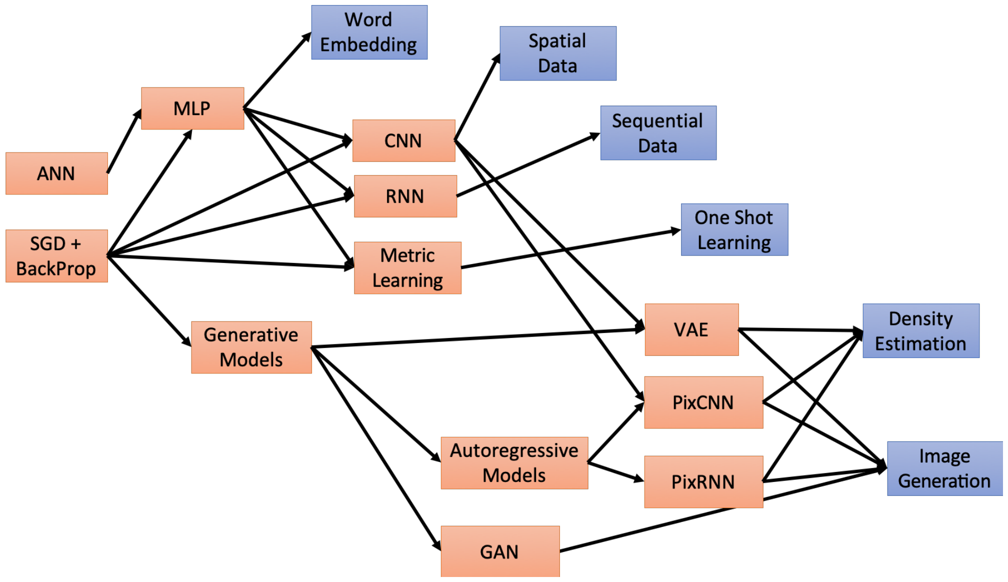


Figure 12: Course overview

The dependencies of the course topics are given in fig. 12. The course is organized in five content chapters:

- Primer of Neural networks
- Word embedding
- Models for spatially distributed data
- Models for sequential data
- Generative models

The assignment of each of the topics to the course chapters is presented in fig. 13

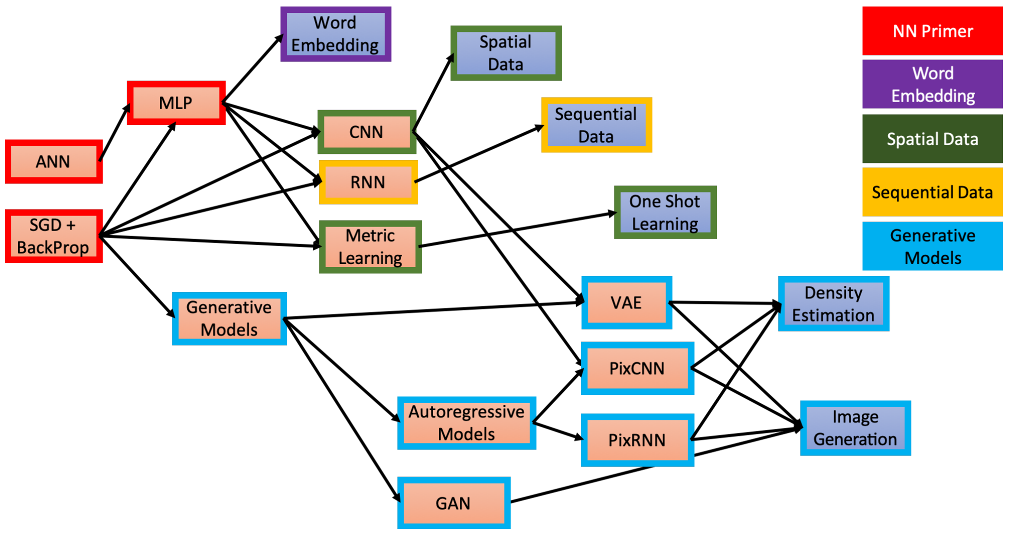


Figure 13: Caption