A

Primer on relevant mathematical notation and concepts

Contents

A.1 Sets and their basic notation 124
A.1.1 Important sets 124
A.1.2 Subsets 125
A.1.3 Set membership 125
A.1.4 Miscellaneous notation 125
A.2 Scalars, vectors, matrices, and tensors 125
A.2.1 Scalars 125
A.2.2 Vectors 126
A.2.3 Matrices 127
A.2.4 Tensors 128
A.3 Sequences 128
A.3.1 Ordered sets 129
A.3.2 Summing sequences 129
A.3.3 Multiplying sequences 129
A.4 Tensor operations 129
A.4.1 Linear combinations 130
A.4.2 Dot product 130
A.4.3 Matrix multiplication 130
A.4.4 Matrix tranpose 131
A.5 Random numbers and variables 131
A.5.1 Uniform distribution 132
A.5.2 Normal distribution 132
A.5.3 Bernoulli distribution 134
A.5.4 Expectation 134
A.5.5 Independent and identically distributed random variables 135

A.6 Basic calculus **136**
A.6.1 Differentiation: computing a gradient 136
A.6.2 Integration: computing an integral 138
A.7 Notation for Convolutional Neural Networks (CNNs) **140**
A.7.1 Partial networks 141
A.7.2 Inputs, outputs, and intermediate tensors 141

In this primer, we describe essential mathematical notation and concepts used in this thesis. Note that notation might be slightly different in the chapters presented as papers (*i.e.* chapter 2 uses the notation presented here). Nevertheless, the concepts described here are highly relevant to understanding the material in this thesis and any other material that discusses CNNs.

This primer only assumes basic algebra (*e.g.* comfort with using variables) and is heavily based on Quantstart, 2017 and Brownlee, 2018.

A.1 Sets and their basic notation

A **set** is an *unordered*[1] collection of objects (*i.e.* order does not matter) with no duplicates. It can be either *finite* or *infinite*[2] in size. For example, the following are sets of the primary colors, a few points, and the natural numbers (*i.e.* set of positive integers) respectively:

$$\begin{aligned} A =& \{\text{red}, \text{blue}, \text{yellow}\}, \\ B =& \{(0,0), (0,1), (1,0), (1,1)\}, \\ C =& \{1, 2, 3, 4, \ldots\}. \end{aligned} \tag{A.1}$$

Sets are most frequently described using curly braces (*i.e.* $\{\ldots\}$) with elements delineated by commas and are often represented as uppercase, italicized variables (*i.e.* A, B, C), though they can be represented by other styles of uppercase variables.

A.1.1 Important sets

There are a few important sets that are represented by specific symbols. We have already described the set of natural numbers, which is denoted by $\mathbb{N} = \{1, 2, 3, \ldots\}$. Then there is the set of all integers (*i.e.* positive, negative, and zero), which is denoted by $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$. Finally, there is the set of real numbers (*i.e.* all numbers that lie on a number line), which is given by $\mathbb{R}$; this includes

[1] We introduce ordered sets later in this section; assume sets are unordered by default.
[2] *i.e.* there can be an infinite number of objects in a set.

numbers like -1 (*i.e.* integers), $\frac{1}{3}$ (*i.e.* rational numbers that can be represented as a fraction), and π (*i.e.* irrational numbers).

A.1.2 Subsets

A **subset**[3] is a set which is entirely contained in another set and is denoted as follows:

$$A \subseteq B, \tag{A.2}$$

where $\subseteq$ can be read as "is a subset of". For instance, $\mathbb{N} \subseteq \mathbb{Z}$ but $\mathbb{N} \not\subseteq \mathbb{Z}$ (where $\not\subseteq$ reads as "is not a subset of"), because the natural numbers (*i.e.* positive integers) is a subset of all integers but the reverse relationship is not true.

There are several short-hand ways to describe subsets of real numbers (*i.e.* $\mathbb{R}$) that start and end at specific numbers. Consider all numbers between 0 and 1. If we wanted to describe this set of numbers and include 0 and 1, we would describe it using the following notation: $[0, 1]$, where square brackets denotes inclusion. If we wanted to exclude 0 and 1 from the set, we would describe it as follows: $(0, 1)$, where parentheses denotes exclusion. If we wanted to include 0 but exclude 1 from the set, we would describe it as follows: $[0, 1)$. Thus, the set of real numbers between a and b can be described using square brackets and/or parentheses, depending on whether each endpoint should be included or excluded from the set.

A.1.3 Set membership

To denote that an object can be found in a set, we use the inclusion symbol: $\in$. For example, $x \in \mathbb{R}$ states that x is in the set of real numbers (*i.e.* it is a real number).

A.1.4 Miscellaneous notation

To describe a statement that holds **for all** elements in a set, we use the following notation: $\forall x \in S \ldots$, which reads as "for all elements x in set S" as $\forall$ denotes "for all."

A.2 Scalars, vectors, matrices, and tensors

A.2.1 Scalars

A **scalar** is a single number and is represented by lowercase, italicized variables, such as $x = 1.2$ and $y = -2$. The important sets mentioned earlier (*i.e.* $\mathbb{R}$) are all sets of scalars.

[3] More precisely, A is a subset of B if and only if all elements in A are also in B.

A.2.2 Vectors

A **vector** is an *ordered* list of scalars (*a.k.a.* an *array*[4] of scalars). The most common vectors are points on an xy coordinate graph, such as $(1, 2)$; here, it's clear that position matters, as $(1, 2) \neq (2, 1)$. Vectors are represented by lowercase, boldface and italicized variables, such as $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\boldsymbol{x} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}. \tag{A.3}$$

They can be represented as row vectors (*i.e.* a single row, like $\boldsymbol{x}$) or as column vectors (*i.e.* a single column, like $\boldsymbol{y}$) and with square brackets (*i.e.* $[\cdots]$) or parentheses (*i.e.* $(\cdots)$). In this thesis, we will use row and column vectors and both kinds of brackets interchangeably.

We can also describe them using set notation as follows: $\boldsymbol{x} \in \mathbb{N}^4, \boldsymbol{y} \in \mathbb{N}^3$. This denotes that $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors with 4 and 3 elements respectively and are comprised of natural numbers. We note the size of a vector by saying it is "n-dimensional" or of length n, where n denotes the number of elements in a vector — *i.e.* $\boldsymbol{x}$ is a 4-dimensional vector.

To reference an element at a specific position within a vector, we use scalar notation (*i.e.* lowercase, non-boldface) with a subscript to indicate the index of the position:

$$\boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_n \end{bmatrix}. \tag{A.4}$$

In the above example, z_i refers to the element in the i-th position. In the earlier examples, $x_3 = 3$ and $y_2 = 2$.

A.2.2.1 Vector arithmetic

To add two vectors, they must be the same size. Then, for every position in the vector, the two corresponding elements are added together. Given two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^N$, adding the two vectors would result in the following vector $\boldsymbol{c}$:

$$\boldsymbol{c} = \boldsymbol{a} + \boldsymbol{b} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \cdots \\ a_n + b_n \end{bmatrix}. \tag{A.5}$$

[4] In computer science, an *array* is a fixed-length, ordered list.

Similarly, *element-wise multiplication* takes two vectors of the same size and multiplies together each of their corresponding elements to produce another vector of the same size. Given two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^N$, we denote element-wise multiplication using the Hadamard operator $\odot$ as follows:

$$\boldsymbol{c} = \boldsymbol{a} \odot \boldsymbol{b} = \begin{bmatrix} a_1 \times b_1 \\ a_2 \times b_2 \\ \cdots \\ a_n \times b_n \end{bmatrix} \tag{A.6}$$

The resulting vector is also known as the *Hadamard product*.

To multiply a vector with a scalar every element in the vector is multiplied by the scalar. Given a vector $\boldsymbol{c} \in \mathbb{R}^N$ and a scalar $s \in \mathbb{R}$, multiplying (*i.e.* scaling) $\boldsymbol{c}$ by s sould result in the following vector $\boldsymbol{d}$:

$$\boldsymbol{d} = s\boldsymbol{c} = \begin{bmatrix} s \times c_1 \\ s \times c_2 \\ \cdots \\ s \times c_n \end{bmatrix}. \tag{A.7}$$

A.2.3 Matrices

A matrix is a 2-dimensional[5] rectangular array of scalars and represented by uppercase and boldface variables, such as $\mathbf{X}$ and $\mathbf{Y}$:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}. \tag{A.8}$$

Using set notation, we would say $\mathbf{X} \in \mathbb{N}^{2\times3}$ and $\mathbf{Y} \in \mathbb{N}^{3\times2}$, where the superscript $M \times N$ denotes a matrix with M rows and N columns.

To reference elements within a matrix, we use scalar notation (*i.e.* lowercase, non-boldface) with subscripts denoting the row and column of the element:

$$\mathbf{Z} = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,n} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{m,1} & z_{m,2} & \cdots & z_{m,n} \end{bmatrix}. \tag{A.9}$$

In the above example, $z_{i,j}$ refers to the element at in the i-th row and j-th column of the matrix $\mathbf{Z} \in \mathbb{R}^{M\times N}$. In the earlier examples, $x_{2,1} = 4$ and $y_{2,1} = 3$.

[5] The *dimensionality* of an array denotes its geometric shape. A vector is a 1-dimensional array because it expands *along* one dimension (*i.e.* either down a column vector or across a row vector). This is in contrast to the dimensionality of a vector, which denotes the number of elements in it.

A.2.4 Tensors

A **tensor** is a generalization of scalars, vectors, and matrices and can be an array of any *dimensionality*.[6] In the context of CNNs, 3D and 4D tensors are often used. An image can be viewed as a 3D tensor, where the 3rd dimension corresponds to the color channel. A batch of images (*i.e.* ordered set of images) can be viewed as a 4D tensor, where the 4th dimension corresponds to the index of an image.

In this thesis, we represent tensors such as an image tensor as follows: $\boldsymbol{x} \in \mathbb{R}^{3\times H\times W}$, where the tensor variable $\boldsymbol{x}$ is represented in the same way as a vector (*i.e.* lowercase, boldface, italicized) and the number of elements being multiplied in the superscript of a set (*i.e.* 3 elements: 3, H, and W) denotes the order of the tensor. Conceptually, a 3D tensor can be imagined as a cube[7] with a specific depth, height, and width. An element in a 3D tensor is given by $x_{i,j,k}$, where i, j, k denotes the index of the channel, row, and column respectively.

We represent a 4D tensor (*e.g.* batch of images) as follows: $\boldsymbol{x} \in \mathbb{R}^{B\times C\times H\times W}$, where B denotes the batch size (*i.e.* number of images), C denotes the number of channels ($C = 3$ for RGB images), H denotes the height (*i.e.* number of rows), and W denotes the width (*i.e.* number of columns). An element in a 4th-order tensor is given by $x_{i,j,k,l}$, where i, j, k, l denotes the index of the batch element, channel, row, and column respectively.

In this thesis, we will always order the dimensions of tensors with order greater than 1 as follows: 1. batch (if given); 2. channel (if given); 3. row; 4. column.

Arithmetic operations on tensors (*e.g.* addition, element-wise multiplication, scalar multiplication) work in the same way as those on vectors; matrix multiplication can also be generalized to larger-order tensors.

A.3 Sequences

Mathematical operators are often applied to sequences (*i.e.* sum up all the elements in a vector). In this section, we describe common notation used to succintly describe sequences.

[6] The *dimensionality* of an array is also known as the *order* of an array — *i.e.* a 3D tensor can also be described as a 3rd-order tensor.

[7] More precisely, as a *cuboid*.

A.3.1 Ordered sets

To describe an *ordered set*[8] with N elements, we use the following notation:

$$\{x_i\}_{i=1}^{N} = \{x_1, x_2, \ldots, x_{N-1}, x_N\}, \tag{A.10}$$

where x_i refers to the i-th element in the set, and the sub and super-script detail the variable representing the index (*i.e.* i), its starting value (*i.e.* 1) and its ending value (*i.e.* N). An ordered set can contain duplicates, that is, $x_i = x_j$ where $i \neq j$ is allowed. In this thesis, assume that sets are unordered by default, unless explicitly stated or denoted with this index notation.

A.3.2 Summing sequences

To describe summing along a sequence, we use the following notation:

$$y = \sum_{i=1}^{N} x_i = x_1 + x_2 + \ldots + x_{n-1} + x_n, \tag{A.11}$$

where x_i denotes the i-the element in the sequence and the uppercase Greek letter Sigma (*i.e.* Σ) represents the sum operation.

A.3.3 Multiplying sequences

To describe taking the product (*i.e.* multiplying together) of elements of a sequence, we use the following notation:

$$y = \prod_{i=1}^{N} x_i = x_1 \times x_2 \times \ldots x_{n-1} \times x_n, \tag{A.12}$$

where the uppercase Greek letter Pi (Π) represents the product operation.

A.4 Tensor operations

In the context of deep learning, we often describe custom operations to tensors (*i.e.* passing a tensor through a layer). We use function notation to describe operations: $f : D \to R$, where f denotes the function (*i.e.* operation), D denotes the *domain* (*i.e.* the set of possible input values), and R denotes the *range* (*i.e.* the set of possible output values).

[8] For the sake of simplicity, we abuse notation and describe what is typically known as a *indexed family* as an *ordered set* — *i.e.* a collection of objects in which each object has an index. See the Wikiepedia article on "Indexed family."

The operations we consider typically take one input and produce one output (though some CNN operations can take multiple inputs and produce multiple outputs). We will most often describe this as follows:

$$\boldsymbol{y} = f(\boldsymbol{x}), \tag{A.13}$$

where $\boldsymbol{x}$ denotes the input tensor the operation f and $\boldsymbol{y}$ denotes the output of operation f when applied to $\boldsymbol{x}$.

A.4.1 Linear combinations

A common operation that is frequently used in neural networks is computing the **linear combination** of terms. A linear combination refers to the operation that, when given a set of tensors (*i.e.* $\{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$) and a set of scalars (*i.e.* $\{a, b, c\}$), multiplies each term with a constant and sums the result:

$$a\boldsymbol{x} + b\boldsymbol{y} + c\boldsymbol{z}, \tag{A.14}$$

In the context of deep neural networks, the terms typically correspond to the values in an input tensor and the constants to the learned parameters (*i.e.* weights).

A.4.2 Dot product

Another common operation that is closely related to linear combinations is taking the **dot product** between two tensors (*i.e.* $\boldsymbol{x}$ and $\boldsymbol{y}$) of the same size. The dot product is the sum of the products of the corresponding entries in two sequences (*i.e.* the Hadamard product) and is denoted using the $\cdot$ operator. For example, the dot product of two 3D tensors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{C \times H \times W}$ is given by

$$z = \boldsymbol{x} \cdot \boldsymbol{y} = \sum \boldsymbol{x} \odot \boldsymbol{y} = \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{c,i,j} \cdot y_{c,i,j}, \tag{A.15}$$

In the context of convolutional weights, an output value is computed by taking the dot product of a set of weights and a subset of input values of the same size.

A.4.3 Matrix multiplication

Matrix multiplication is also frequently used in deep learning. In order to do multiply two matrices, $\mathbf{A} \in \mathbb{R}^{M \times N}$ and $\mathbf{B} \in \mathbb{R}^{N \times P}$, the number of columns in A needs to equal the number of rows in $\mathbf{B}$ (*i.e.* N, the inner dimensions need to match). The dimensions of resulting matrix $\mathbf{C} \in \mathbb{R}^{M \times P}$ is given by the outer

dimensions of the two matrices (*i.e.* number of rows in $\mathbf{A}$ and number of columns in $\mathbf{B}$). Then, an element in the resulting matrix $\mathbf{C} = \mathbf{AB}$ is computed as the dot product of a row in $\mathbf{A}$ and a column in $\mathbf{B}$:

$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} \cdot b_{k,j} \tag{A.16}$$

Note that, by definition, the order of the matrices being multiplied matters, that is $\mathbf{AB} \neq \mathbf{BA}$. Here are the matrices with their element notation:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix} \begin{bmatrix} b_{1,1} & \dots & b_{1,p} \\ b_{2,1} & \dots & b_{2,p} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \dots & b_{m,p} \end{bmatrix} = \begin{bmatrix} c_{1,1} & \dots & c_{1,p} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \dots & c_{m,p} \end{bmatrix}. \tag{A.17}$$

Now, we can work out the following example. Given the following matrices,

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{bmatrix} \tag{A.18}$$

The result of multiplying the matrices $\mathbf{C} = \mathbf{AB}$ is given as

$$\mathbf{C} = \begin{bmatrix} 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 3 & 1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 \\ 2 \cdot 1 + 3 \cdot 2 + 4 \cdot 3 & 2 \cdot 2 + 3 \cdot 3 + 4 \cdot 4 \end{bmatrix} = \begin{bmatrix} 14 & 20 \\ 20 & 29 \end{bmatrix} \tag{A.19}$$

A.4.4 Matrix tranpose

Often, the **transpose** of a matrix is taken before doing matrix multiplication in order to make the inner dimensions align. A matrix transpose, denoted as $\mathbf{M}^T \in \mathbb{R}^{N \times M}$ is a version of another matrix $\mathbf{M} \in \mathbb{R}^{M \times N}$ in which its rows and columns have been swapped.

Here is an example of a matrix and its transpose:

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \mathbf{M}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}. \tag{A.20}$$

A.5 Random numbers and variables

For a more detailed treatment on basic probability, see Blitzstein et al., 2019.

Many times, we desire to generate a random number. There are two main *distributions*[9] from which we typically *sample* (*i.e.* draw or select) random numbers.

[9] A *distribution* is a succinct description of all possible elements and the likelihoods (*i.e.* probabilities) with which they can be sampled.

Basic notation. We represent a random number and the distribution it is drawn from as follows:

$$X \sim \mathcal{D}, \tag{A.21}$$

where $\sim$ represents drawing X (on its left-hand) from $\mathcal{D}$ (on its right-hand), X denotes a random variable that represents the random number drawn, and $\mathcal{D}$ represents the distribution being drawn from.

A.5.1 Uniform distribution

The **uniform distribution** is one of the most basic random distribution and describes drawing with equal likelihood (*i.e.* "drawing uniformly") any option from a set of possible options.

We most typically deal with a uniform distribution of a subset of the real numbers (*i.e.* $\mathbb{R}$) and can describe it in the same way we describe a range of real-numbers starting at the minimal point a and ending at the maximal point b, with the option of including or excluding the endpoints respectively: $[a, b]$ or (a, b).

To describe the Uniform distribution succinctly, we use the following notation:

$$\text{Uniform}(a, b) \text{ or } \mathcal{U}(0, 1), \tag{A.22}$$

where kind of bracket used denotes inclusion or exclusion of the endpoints.

Then, the following denotes a random variable X drawn from $[0, 1]$: $X \sim \mathcal{U}[0, 1]$.

One way to describe the a distribution is via a special function, known as a **probability density function (PDF)**, whose output describes the relative likelihood that the value of a random variable drawn from said distribution would be equal to the input value to the function.

For the uniform distribution, its PDF is given as follows:

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \leq x \leq b, \textit{and} \\ 0, & \text{for } x < a \text{ or } x > b. \end{cases} \tag{A.23}$$

This PDF function shows that all possibilities between a and b are equally likely.

A.5.2 Normal distribution

The **normal distribution** (*a.k.a.* **gaussian** distribution) is another basic random distribution and is most succinctly preciesly described by its PDF function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}, \tag{A.24}$$

where $\mu \in \mathbb{R}$ (lowercase Greek letter mu) is a real number that describes the *mean* (*i.e.* average) of the distribution and $\sigma \in \mathbb{R}^+$ (lowercase Greek letter sigma) is a positive, real number that describes the *standard deviation* of the distribution.

A plot of the PDF function $f(x)$ looks like a bell-shaped curve, where the middle of the bell is at mean μ and the standard deviation σ controls how spread out the bell-shape curve is.

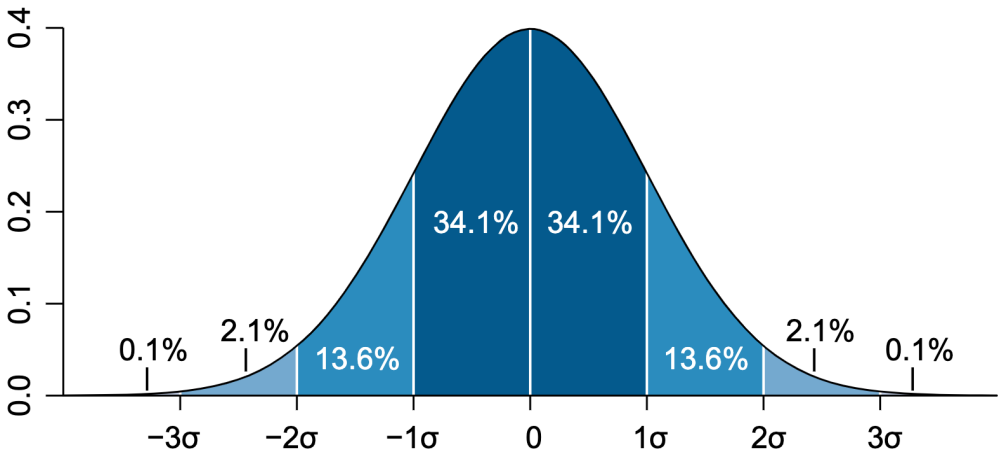

Figure A.1: PDF of normal distribution with $\mu = 0$. The percentages shown highlight the percentage of the total area under the curve that is contained in a given portion of the curve. Notice how these percentages demonstrate the 68-95-99.7 rule. Reproduced image by M. W. Toews under CC BY 2.5.

More precisely, the standard deviation σ is the width on both sides of the mean that defines an input range of $(\mu - \sigma, \mu + \sigma)$. The portion of the area under the curve that falls within that input range (*i.e.* within one standard deviation, $\pm\sigma$) is 68% of the total *area under the curve* (AUC). The corresponding portion that falls within two standard deviations (*i.e.* input range of $(\mu - 2\sigma, \mu + 2\sigma)$) is 95% of the total AUC, and the corresponding portion for 3 standard deviations (*i.e.* input range of $(\mu - 3\sigma, \mu + 3\sigma)$) is 99.7%. This property of normal distributions holds regardless of specific μ and σ values and is known as the *68-95-99.7 rule*.

Intuitively, samples drawn from a normal distribution are more likely to be close to its mean μ than far away from it, with values greater than 3 standard deviations (*i.e.* σ) away less than 0.3% likely to be drawn.

Notation describing the normal distribution is as follows

$$\text{Normal}(\mu, \sigma) \text{ or } \mathcal{N}(\mu, \sigma), \tag{A.25}$$

where μ and σ denote its mean and standard deviation respectively.

Standard normal distribution. Also known as the **unit normal distribution**, the **standard normal distribution** is the Normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$ — *i.e.* $\mathcal{N}(0, 1)$) — and is frequently used, hence its special designation.

A.5.3 Bernoulli distribution

So far, we have only discussed *continuous* distributions that span a range of real numbers. However, we can also have *discrete distributions* which present a set of categories each with a different likelihood of getting drawn. Let us consider the simplest discrete distribution: the **Bernoulli distribution**. Intuitively, the Bernoulli distribution can be likened to a heads or tails coin flip of a weighted coin. Let p be the probability of seeing heads; then, the probability of seeing tails is $1 - p$. Formally, if X is a Bernoulli random variable with parameter p, then

$$P(X = 1) = p; P(X = 0) = 1 - p, \tag{A.26}$$

where $X = 1$ corresponds to the "heads" outcome in the coin flip analogy. Then, a fair coin could be represented as a Bernoulli with $p = 0.5$.

A.5.4 Expectation

An high-level understanding how to compute an expectation is helpful for understanding the expected gradients method (Sturmfels et al., 2020), which we discuss in section 2.2.1.2 as part of our literature review. In this section, we provide such a high-level explanation.[10]

The **expectation** of a random variable is the weighted average of all possible options. Intuitively, it is the arithmetic mean of a distribution.

For a finite, discrete random variable X with a finite set of possibilities: $x_1, x_2, \ldots, x_n$, its expectation is as follows:

$$\mathbb{E}[X] = \sum_{i=1}^{n} x_i P(X = x_i) = x_1 P(X = x_1) + \ldots + x_n P(X = x_n) \tag{A.27}$$

Consider a random variable X that is Bernoulli distribution with parameter p. Using eq. (A.27), we can compute the expectation of a Bernoulli as follows:

$$\mathbb{E}[X] = 0P(X = 0) + 1P(X = 1) = p \tag{A.28}$$

[10] See the Wikipedia article on "Expected value" for more information.

For the other two continuous distributions we covered, we will use the intuitive definition of the arithmetic mean to introduce their expectations. The expectation for a normally distributed random variable is given by μ:

$$\mathbb{E}[X] = \mu. \tag{A.29}$$

For a uniformally distributed random variable $X \sim \text{Uniform}(a, b)$, its expectation is given by

$$\mathbb{E}[X] = \frac{b-a}{2}. \tag{A.30}$$

Intuitively, this makes sense: the mean of a uniformally distributed random variable is the midpoint between the endpoints of the distribution.

A.5.5 Independent and identically distributed random variables

So far, we described drawing real-valued scalars from the uniform and normal distribution. However, in the context of deep learning, we often deal with whole tensors that are randomly generated. The elements of such tensors can be described most commonly as being **independent and identically distributed (i.i.d.)** random variables.

Two random variables X and Y are *identically distributed* if they are drawn from the same distribution (*i.e.* they have the same PDF function). Given the following random variables:

$$X \sim \mathcal{N}(0,1), Y \sim \mathcal{N}(0,1), Z \sim \mathcal{N}(0,2), \tag{A.31}$$

X and Y are identically distributed but X and Z are not (because they have different standard deviations σ, resulting in different PDF functions).

Two random variables X and Y are *independent* if the processes that generate them are indepedent (*e.g.* not conditioned) on each other. For example, snowy conditions and the temperatures are not independent events because snowy conditions necessitates a temperature below freezing point (*i.e.* snow is conditioned on temperature). One way to formally describe independent events is that the probability of both events happening is equal the product of the probability of both events happening on its own. This can be described mathematically as follows:

$$f_{X,Y}(x, y) = f_X(x) f_Y(y), \text{ for all } x, y, \tag{A.32}$$

where f_X and f_Y are the PDFs for random variables X and Y respectively and $f_{X,Y}$ is the joint PDF that describes the likelihoods of how pairs of possibilities (x, y) occur.

In our context, we often sample a tensor whose elements are i.i.d. from the same distribution. We use the following notation to describe this:

$$\boldsymbol{x} \overset{\text{i.i.d.}}{\sim} D, \tag{A.33}$$

where $\boldsymbol{x}$ is a tensor whose elements are i.i.d. sampled from distribution D. Thus, $\boldsymbol{x} \overset{\text{i.i.d.}}{\sim} \mathcal{U}[0, 1]$ denotes a tensor whose elements are all uniformly drawn from the range $[0, 1]$.

A.6 Basic calculus

In this section, we cover some basic concepts from calculus[11] that are helpful for understanding how CNNs work. To read this thesis, one does not need to fully understand these concepts, but simply need to understand at a high-lvel what computing the "gradient" or "integral" means. Thus, we focus on providing an intuititive explanation of these concepts and refer the reader to other sources to fully understand them.

A.6.1 Differentiation: computing a gradient

A **derivative** or **gradient** of a function f at a point x refers to the *slope* (*a.k.a.* the "rate of change") of the function at that point (see fig. A.2).[12]

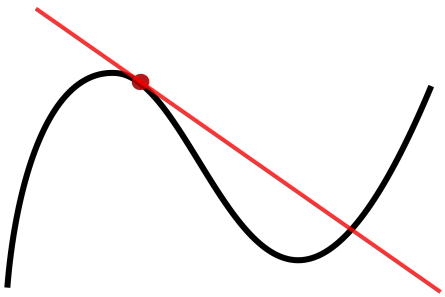

Figure A.2: Gradient of a function. This diagram visualizes a function (black curve) at the specific point (red dot). The gradient of the function at the point is given by the slope of the red line. This image is by Jacj at English Wikipedia and is available in the public domain.

[11] Calculus comes from the Latin word meaning "small stone" and primarily concerns studying how small changes affect a function. See this article on "Introduction to Calculus" for an accessible introduction.

[12] This section is based on this article titled "Introduction to Derivatives."

Slope between two points. On an xy-plot (*i.e.* a Cartesian plot), the slope of a function is expressed as follows (fig. A.3):

$$\text{slope} = \frac{\text{change in } y}{\text{change in } x} = \frac{\Delta y}{\Delta x}, \tag{A.34}$$

where the upper-case Greek symbol delta (Δ) is read as "change in" and represents the difference between two quantities. Formally, the slope m between two points (x_1, y_1) and (x_2, y_2) can be calculated as follows:

$$m = \frac{y_2 - y_1}{x_2 - x_1}. \tag{A.35}$$

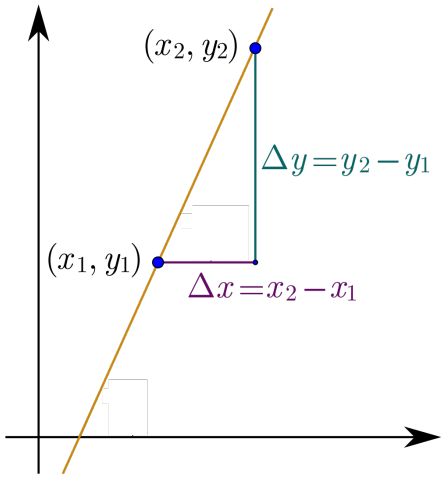

Figure A.3: Slope between two points. The slope between two points can be calculated as the change in y over the change of x — *i.e.* $\Delta y/\Delta x$, see eq. (A.35). This image was originally created by Maschen; we modified and reproduced it via CC BY-SA 3.0.

For a motivational example, consider how we describe the speed of cars. Suppose you drive 20 miles in 20 minutes. We would describe that speed as 60 miles per hour. This is actually a slope and describes the average rate of change (*i.e.* change of miles per hour) of a vehicle and is computed as follows:

$$\frac{20 \text{ miles}}{\frac{1}{3} \text{ hour}} = 60 \text{ mph}. \tag{A.36}$$

Gradient at a single point. To find the slope at a single point, we consider a very small change in x $x + \Delta x$ — affects the output of a scalar function f. We can express the slope between x and $x + \Delta x$ as follows:

$$\frac{\Delta y}{\Delta x} = \frac{f(x + \Delta x) - f(x)}{\Delta x}. \tag{A.37}$$

Then, to find a slope at x, we consider what happens when Δx shrinks to become infinitesimally small, that is, when Δx goes to 0 (*i.e.* $\Delta x \to 0$). This is the gradient or derivative of a function f at point x (see fig. A.4).

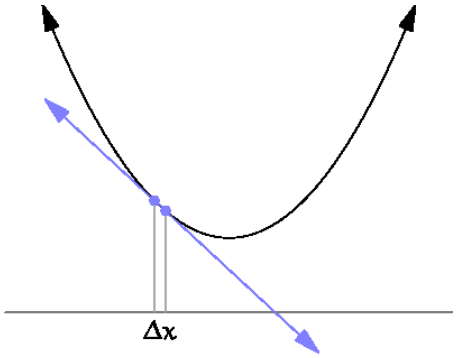

Figure A.4: Δx. Intuitively, the gradient of a function at a point x is the slope of the function between x and $x + \Delta x$ when Δx is infinitesimally small (*i.e.* when Δx goes to 0). This image is part of an animated image originally created by Wikipedia user Brnbrnz and is reproduced it via CC BY-SA 4.0.

Here, we have shown the gradient of a scalar y with respect to another scalar x. In the context of CNNs, a gradient of a scalar is typically computed with respect to a multi-dimensional tensor — *i.e.* the gradient of a loss function with respect to a network's weight parameters, which are typically stored as tensors.

In this thesis, we represent a gradient of a scalar y with respect to a tensor $\boldsymbol{x}$ as follows:

$$\frac{\partial y}{\partial \boldsymbol{x}}, \tag{A.38}$$

where the gradient would have the same shape as $\boldsymbol{x}$.

A.6.2 Integration: computing an integral

An high-level understanding of integrals is helpful for understanding the integrated gradients method (Sundararajan et al., 2017), which we discuss in section 2.2.1.2 as part of our literature review.[13]

Intuitively, an **integral** can be viewed as the "area under a curve," and a *definite integral* is the area under a curve in between points a and b. Figure A.5 visualizes this interpretation. Given a scalar function f (*i.e.* the "curve"), the definite integral of f between a and b is the sum of the colored areas between the function and the x-axis. Areas where $f(x) < 0$ contribute negatively to the integral, while areas where $f(x) > 0$ contribute positively to the integral.

[13] For an accessible introduction to integration, see this article on "Introduction to Integration."

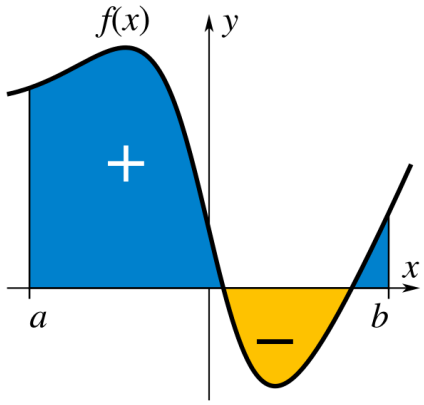

Figure A.5: Example of an integral. The definite integral of a function $f(x)$ between points a and b is the sum of the signed areas under the "curve" of the function. For the parts where $f(x) < 0$ (in yellow), those areas contribute negatively to the integral. This image is by Wikipedia user KSmrq and is reproduced via CC BY-SA 3.0.

One way to think about approximating an integral is to be summing up "slices" underneath a curve. Let us consider integrating the function $f(x) = \sqrt{x}$ between 0 and 1. We can do this by "slicing up" the function as done in fig. A.6. Then,

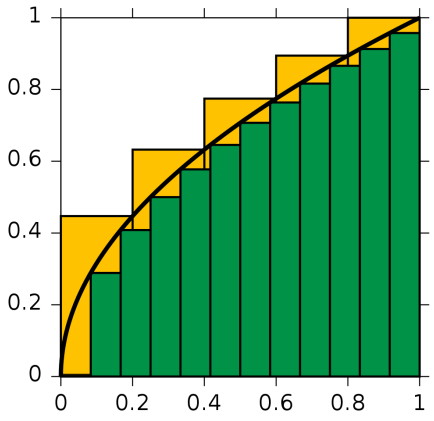

Figure A.6: Approximating an integral. Here is an example of how the integral of the function $f(x) = \sqrt{x}$ from $x = 0$ to $x = 1$ can be approximated. This image is by Wikipedia user KSmrq and is reproduced via CC BY-SA 3.0.

the integral is approximately equal to the sum of the yellow slices (or the sum of the green slices). Concretely, let us work out the example shown in yellow. Here, we evaluate $f(x)$ at the following points: $0.2, 0.4, 0.6, 0.8, 1$ (*i.e.* in increments of $\Delta x = 0.2$). To compute the area of one slice, we multiply the height of the slice (*i.e.* $f(x)$) with its width (*i.e.* Δx). Then, we can approximate the integral

by summing up the areas of all slices:

$$0.2(f(0.2) + f(0.4) + f(0.6) + f(0.8) + f(1.0)) = 0.7497. \tag{A.39}$$

To compute an integral exactly, we consider what would happen if the slices became infinitesimally small (*i.e.* $\Delta x \to 0$). Formally, we write a definite integral as follows:

$$c = \int_a^b f(x)dx, \tag{A.40}$$

where a and b denote where to start and end integrating, $f(x)$ denotes the function to integrate, and dx denotes we are considering slices along x (*i.e.* we start integrating at $x = a$ and stop at $x = b$).

An integral can be viewed as the inverse of the derivative. To illustrate the relationship between an integral and a derivative (*i.e.* a rate of change), let us consider the vehicle speed example again. Let $f(x)$ denote the speed of our vehicle at time x (in hours). Then, if we are interested in the distance we have travelled in the first 1 hour of driving, we can express it as follows: $\int_0^1 f(x)dx$. Suppose we maintained an expect speed of 20 m.p.h for the first 20 minutes, 40 m.p.h. for the second 20 minutes, and 60 m.p.h. for the third 20 minute-segment. Then, we can write $f(x)$ as follows:

$$f(x) = \begin{cases} 20 & \text{for } 0 \leq x \leq \frac{1}{3}, \\ 40 & \text{for } \frac{1}{3} \leq x \leq \frac{2}{3}, \\ 60 & \text{for } \frac{2}{3} \leq x \leq 1. \end{cases} \tag{A.41}$$

Now, we can compute our total distance in an hour as follows:

$$\int_0^1 f(x)dx = \frac{1}{3}(20 + 40 + 60) = 40 \text{ miles}. \tag{A.42}$$

A.7 Notation for Convolutional Neural Networks (CNNs)

In this thesis, we describe an object classification CNN as the following function:

$$\Phi : \mathbb{R}^{3\times H\times W} \to [0, 1]^C, \tag{A.43}$$

where Φ (*i.e.* uppercase Greek letter Phi) denotes the CNN, $\mathbb{R}^{3\times H\times W}$ specifies the input domain as the set of 3D tensors with 3 color channels, H rows, and W columns that contain real numbers (*i.e.* RGB images), and $[0, 1]^C$ denotes the

output range the set of vectors of length C with elements ranging from 0 to 1 inclusive. The output of the CNN can be viewed as vector containing probabilities, where each element represents the probability that a specific object category is the dominant object in the image.

To denote that the values of the network's output vector has not been normalized to range between 0 and 1 (*i.e.* does not represent probabilities), we would describe it as follows:

$$\Phi : \mathbb{R}^{3\times H\times W} \to \mathbb{R}^{C}. \tag{A.44}$$

A.7.1 Partial networks

To describe the set of operations (*i.e.* layers) from one point in the network Φ to another (*i.e.* from layer l_1 exclusive to layer l_2 inclusive) we use the following notation:

$$\Phi_{l_1}^{l_2}, \tag{A.45}$$

where the subscript denotes that the output of layer l_1 is the input to our partial network $\Phi_{l_1}^{l_2}$ and the superscript denotes that l_2 is the last layer applied to produce the output of the partial network.

If we want to describe the network up to a certain point (*i.e.* starting from the first layer to layer l), we use the following notation:

$$\Phi^{l}, \tag{A.46}$$

where the superscript denotes the last layer applied to produce this partial network's output.

Finally, to describe selecting an element from the output tensor produced by the network Φ, we use the following notation:

$$\Phi_c : \mathbb{R}^{3\times H\times W} \to \mathbb{R}, \tag{A.47}$$

where c denotes the index of a particular object category whose output we are interested in.

A.7.2 Inputs, outputs, and intermediate tensors

A CNN can often be described as applying a series of layers (*i.e.* operations) that each take as input a tensor and produce as output another tensor, that may or may not have a different shape (*i.e.* the domain and range of the operation may be different). In this thesis, we will usually describe the input tensor to a CNN Φ as

$\boldsymbol{x}$ and the output tensor produced by Φ as $\hat{\boldsymbol{y}}$, and an intermediate tensor (*i.e.* the output of a layer that is not the last layer in Φ) as $\boldsymbol{z}$. We use the hat symbol on top of $\boldsymbol{y}$ (*i.e.* $\hat{\boldsymbol{y}}$, read as "y-hat") to denote an estimate of $\boldsymbol{y}$ (*i.e.* the target or ground-truth label). Intermediate tensors are also known as **activations** or **activation tensors**.