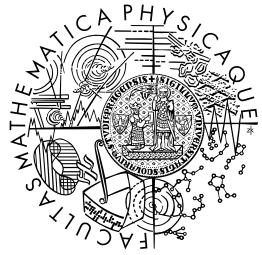

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Martin Grätzer

Neural Networks and Knowledge Distillation

Department of Probability and Mathematical Statistics

Supervisor of the master thesis: Mgr. Ondřej Týbl, Ph.D.

Study programme: Financial and Insurance Mathematics

Prague 2025

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Neural Networks and Knowledge Distillation

Author: Bc. Martin Grätzer

Department: Department of Probability and Mathematical Statistics

Supervisor: Mgr. Ondřej Týbl, Ph.D., Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague

Abstract: Abstract.

Keywords: neural networks machine learning knowledge distillation KL divergence

Název práce: Neuronové sítě a destilace znalostí

Autor: Bc. Martin Grätzer

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce: Mgr. Ondřej Týbl, Ph.D., Katedra kybernetiky, Fakulta elektrotechnická, České vysoké učení technické v Praze

Abstrakt: Abstrakt práce přeložte také do češtiny.

Klíčová slova: neuronové sítě, strojové učení, destilace znalostí, KL divergence

Contents

Introduction

In the recent years we have experienced a remarkable surge in artificial intelligence (AI). This rise has been fueled by an increase in computational power, making the creation of more powerful and complex models feasible. However, when deploying a model to a large number of users, we are usually more stringent regarding latency, as well as computational and storage capacity. Yet, simply using a smaller model does not take full advantage of the training capacity we usually possess.

A proposed solution to these seemingly opposing constrains is knowledge distillation. This approach involves training a large model, known as the teacher, and transferring its knowledge to a smaller model, called student, we want to deploy. We believe that the teacher is able to better extract the structure from the data. It learns to differentiate between large number of classes and then correctly predict the label when exposed to new data. Additionally, the trained model also assigns weights to all of the possible classes, which are then converted into probabilities using a softmax function. Even though these are often very small for the incorrect answers, they can still provide valuable information about how the larger model generalizes.

For example, an image of a horse will be correctly labeled by the teacher model with high probability close to 1. However, the model might also assign a small but nonzero probability that the image is a zebra. We argue that this probability will still be many times higher than the probability assigned to an unrelated class, such as a car.

Transferring this knowledge from the teacher to the student is done through distillation, where the student model is trained using the class probabilities produced by the teacher as soft targets. In the original paper, the distillation process is formulated as the minimization of the Kullback–Leibler (KL) divergence.

In this work, we propose enhancing the distillation process by replacing the KL divergence with Rényi divergence, which serves as its generalization, and introduces an additional hyperparameter α. We aim to formally define this new distillation framework, analyze the theoretical properties of Rényi-based distillation, and conduct experiments to evaluate the appropriateness of this approach.

1. Rényi Divergence and Knowledge Distillation

In this chapter, we begin by examining the concepts of entropy, cross-entropy, and divergence. In particular, we define Rényi divergence, establish its connection to KL divergence, and inspect some of the theoretical properties stated in van Erven and Harremoës [2012]. In the second part, we formally define the notion of knowledge distillation, as proposed in Hinton et al. [2015], and inspect some of the theoretical results presented therein. Furthermore, we analyze how these results change when incorporating Rényi divergence into the distillation process.

1.1 KL Divergence and Rényi Divergence

The concept of entropy, as the amount of uncertainty regarding the outcome of an experiment, was introduced by Shannon [1948].

Definition 1. *The entropy of a probability distribution $P = (p_1, \dots, p_n)$ is given by*

$$H(P) = -\sum_{i=1}^{n} p_i \log p_i,$$

where we adopt the convention that $0 \log 0 = 0$.

Example. Let P be the probability distribution of a fair coin toss, i.e., $P = (\frac{1}{2}, \frac{1}{2})$. The entropy $H(P)$ is approximately 0.693. Next, let Q represent the probability distribution of an unfair coin toss, i.e., $Q = (\frac{4}{10}, \frac{6}{10})$. Here, the entropy $H(Q)$ is smaller than $H(P)$, approximately 0.673. In other words, we are less uncertain about the outcome of the unfair coin toss than about the fair coin toss.

To determine the similarity between two probability distributions, we cannot simply subtract their entropies. For example, the entropy of $P_1 = (\frac{4}{10}, \frac{6}{10})$ is the same as the entropy of $P_2 = (\frac{6}{10}, \frac{4}{10})$, yet they represent different distributions. Therefore, we use the concept of divergence, as proposed by Kullback and Leibler [1951]. First, we define a related notion of cross-entropy.

Definition 2. *The cross-entropy of a probability distribution $P = (p_1, \dots, p_n)$ relative to another distribution $Q = (q_1, \dots, q_n)$ is given by*

$$H(P, Q) = -\sum_{i=1}^{n} p_i \log q_i,$$

where we adopt the convention that $0 \log 0 = 0$.

Example. Let P and Q be the probability distributions as described in the example above, i.e., $P = (\frac{1}{2}, \frac{1}{2})$ and $Q = (\frac{4}{10}, \frac{6}{10})$. The cross-entropy of P relative to Q is $H(P, Q) \approx 0.714$. On the other hand $H(Q, P) \approx 0.693$ and we observe that cross-entropy is not symmetric in its arguments.

Definition 3. *The Kullback–Leibler divergence (KL divergence) of a probability distribution $P = (p_1, \ldots, p_n)$ relative to another distribution $Q = (q_1, \ldots, q_n)$ is given by*

$$D_{KL}(P\|Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i},$$

where we adopt the convention that $\frac{0}{0} = 0$ and $\frac{x}{0} = \infty$ for $x > 0$.

We can decompose the KL divergence into two terms, as

$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \\ &= \sum_{i=1}^{n} p_i \log p_i + \left(- \sum_{i=1}^{n} p_i \log q_i \right), \\ &= - H(P) + H(P, Q) \end{aligned} \tag{1.1}$$

and we observe that cross-entropy can be decomposed into entropy and KL divergence.

Example. Let P and Q be probability distributions, given as $P = (\frac{1}{2}, \frac{1}{2})$ and $Q = (\frac{4}{10}, \frac{6}{10})$. From the previous examples, we know that $H(P) \approx 0.693$ and $H(P, Q) \approx 0.714$. Using Equation 1.1, we can calculate the KL divergence of P relative to Q as $D_{\mathrm{KL}}(P\|Q) = -H(P) + H(P, Q) \approx 0.021$.

Kullback–Leibler divergence was later generalized by Rényi [1961]. We begin with the definition.

Definition 4. *The Rényi divergence of order α of a probability distribution $P = (p_1, \ldots, p_n)$ relative to another distribution $Q = (q_1, \ldots, q_n)$ is given by*

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \sum_{i=1}^{n} p_i^\alpha q_i^{1-\alpha},$$

where α is positive number distinct from 1, and we adopt the convention that $\frac{0}{0} = 0$ and $\frac{x}{0} = \infty$ for $x > 0$.

This definition of Rényi divergence assumes that probability distributions P and Q are discrete. For continuous spaces we can substitute the sum by Lebesgue integral (see van Erven and Harremoës [2012]). Now, we present an example that motivated the introduction of the normalization term $\frac{1}{\alpha - 1}$ in the definition.

Example. Let Q be a probability distribution and A be a set, such that $Q(A) > 0$. Define P as the conditional distribution of Q given A, i.e. $P(x) = Q(x|A) = \frac{Q(x)}{Q(A)}$, for $x \in A$. Now compute the Rényi divergence of P relative to Q

$$
\begin{aligned}
D_\alpha(P\|Q) &= \frac{1}{\alpha-1}\log\sum_{x\in A} P(x)^\alpha Q(x)^{1-\alpha},\\
&= \frac{1}{\alpha-1}\log\sum_{x\in A}\left(\frac{Q(x)}{Q(A)}\right)^\alpha Q(x)^{1-\alpha},\\
&= \frac{1}{\alpha-1}\log\sum_{x\in A}\frac{Q(x)}{Q(A)^\alpha},\\
&= \frac{1}{\alpha-1}\log\left(Q(A)^{-\alpha}\sum_{x\in A}Q(x)\right),\\
&= \frac{1}{\alpha-1}\log Q(A)^{1-\alpha},\\
&= -\log Q(A).
\end{aligned}
$$

In this particular example we observe that the factor $\frac{1}{\alpha-1}$ in the definition Rényi divergence has the effect that $D_\alpha(P\|Q)$ does not depend on α in this example. This factor is moreover crucial in the following consideration.

Definition 4 was formulated for orders $\alpha \in (0,1)\cup(1,\infty)$. We now show that the limits on the borders of the domain for α exist and therefore Rényi divergence can be naturally extended to the cases $\alpha = 0, 1, \infty$. That is, we inspect the limits

$$
\begin{aligned}
D_0(P\|Q) &= \lim_{\alpha\to 0+} D_\alpha(P\|Q),\\
D_1(P\|Q) &= \lim_{\alpha\to 1} D_\alpha(P\|Q),\\
D_\infty(P\|Q) &= \lim_{\alpha\to\infty} D_\alpha(P\|Q).
\end{aligned}
$$

where P and Q are discrete distributions on $\{1,\cdots,n\}$. For $\alpha = 0$, we have

$$
\begin{aligned}
\lim_{\alpha\to 0+} D_\alpha(P\|Q) &= \lim_{\alpha\to 0+}\frac{1}{\alpha-1}\log\sum_{i=1}^n p_i^\alpha q_i^{1-\alpha},\\
&= -\log\sum_{i=1}^n \lim_{\alpha\to 0+} p_i^\alpha q_i^{1-\alpha},\\
&= -\log\sum_{i=1}^n q_i \lim_{\alpha\to 0+} p_i^\alpha\\
&= -\log\sum_{i=1}^n q_i\,\mathbb{1}\{p_i>0\},
\end{aligned}
\tag{1.2}
$$

where $\mathbb{1}$ is the indicator function. For $\alpha = 1$, the limit

$$
\lim_{\alpha\to 1}\frac{1}{\alpha-1}\log\sum_{i=1}^n p_i^\alpha q_i^{1-\alpha}
$$

is of an indeterminate form $\dfrac{0}{0}$, allowing us to apply L'Hopital's Rule and we obtain

$$\begin{aligned}\lim_{\alpha\to 1}\frac{1}{\alpha-1}\log\sum_{i=1}^{n}p_i^{\alpha}q_i^{1-\alpha} &= \lim_{\alpha\to 1}\frac{\sum_{i=1}^{n}p_i^{\alpha}q_i^{1-\alpha}\log p_i - p_i^{\alpha}q_i^{1-\alpha}\log q_i}{\sum_{i=1}^{n}p_i^{\alpha}q_i^{1-\alpha}},\\ &= \frac{\sum_{i=1}^{n}p_i\log p_i - p_i\log q_i}{\sum_{i=1}^{n}p_i},\\ &= \sum_{i=1}^{n}p_i\log\frac{p_i}{q_i}.\end{aligned} \tag{1.3}$$

Lastly, for $\alpha=\infty$, we denote $Z(\alpha)=\sum_{i=1}^{n}p_i^{\alpha}q_i^{1-\alpha}$, and $M=\max_i\frac{p_i}{q_i}$ and let $j\in\{1,\cdots,n\}$ be the first index at which this maximum is attained. We have

$$M^{\alpha}q_j \le Z(\alpha) \le M^{\alpha}\sum_{i=1}^{n}q_i. \tag{1.4}$$

Taking the logarithm and dividing by $\alpha-1$ preserves the inequalities, as the logarithm is a monotonic function and $\alpha>1$. From (1.4), we obtain that

$$\frac{\alpha log M+\log q_j}{\alpha-1} \le \frac{1}{\alpha-1}\log Z(\alpha) \le \frac{\alpha log M+\log 1}{\alpha-1}.$$

Taking the limit $\alpha\to\infty$

$$\log M=\lim_{\alpha\to\infty}\frac{\alpha log M+\log q_j}{\alpha-1} \le \lim_{\alpha\to\infty}\frac{1}{\alpha-1}\log Z(\alpha) \le \lim_{\alpha\to\infty}\frac{\alpha log M+\log 1}{\alpha-1}=\log M.$$

Thus,

$$\begin{aligned}\lim_{\alpha\to\infty}\frac{1}{\alpha-1}\log\sum_{i=1}^{n}p_i^{\alpha}q_i^{1-\alpha} = \lim_{\alpha\to\infty}\frac{1}{\alpha-1}\log Z(\alpha) &= \log M,\\ &= \max_i\log\frac{p_i}{q_i}.\end{aligned} \tag{1.5}$$

The limits (1.2), (1.3) , (1.5) allow us to define the Rényi divergences

$$\begin{aligned}D_0(P\|Q) &= -\log\sum_{i=1}^{n}q_i\,\mathbb{1}\{p_i>0\},\\ D_1(P\|Q) &= \sum_{i=1}^{n}p_i\log\frac{p_i}{q_i},\\ D_\infty(P\|Q) &= \max_i\log\frac{p_i}{q_i}.\end{aligned}$$

Comparing to Definition 3, we see that

$$D_1(P\|Q)=D_{\mathrm{KL}}(P\|Q),$$

and Rényi divergence indeed generalizes KL divergence.

Another important case of Rényi divergence is for $\alpha=\frac{1}{2}$. For only this value the Rényi divergence is symmetric, i.e., $D_{1/2}(P\|Q)=D_{1/2}(Q\|P)$. Even with this additional property, it still does not satisfy the definition of a metric, as the triangle inequality does not hold. However, Rényi divergence of order $\frac{1}{2}$ can be

rewritten as a function of the squared Hellinger distance, which, as defined in Cheng et al. [2024], for discrete probability distributions P and Q is given by

$$H^2(P\|Q) = \frac{1}{2}\sum_{i=1}^{n}(p_i^{\frac{1}{2}} - q_i^{\frac{1}{2}})^2.$$

We also get a relation

$$\frac{1}{2}\sum_{i=1}^{n}(p_i^{\frac{1}{2}} - q_i^{\frac{1}{2}})^2 = \frac{1}{2}\left(\sum_{i=1}^{n}p_i + \sum_{i=1}^{n}q_i - 2\sum_{i=1}^{n}p_i^{\frac{1}{2}}q_i^{\frac{1}{2}}\right) = 1 - \sum_{i=1}^{n}p_i^{\frac{1}{2}}q_i^{\frac{1}{2}},$$

which we can use in the definition of Rényi divergence of order $\frac{1}{2}$ to express it in terms of the Hellinger distance

$$D_{1/2}(P\|Q) = \frac{1}{\frac{1}{2}-1}\log\sum_{i=1}^{n}p_i^{\frac{1}{2}}q_i^{1-\frac{1}{2}} = -2\log(1 - H^2(P\|Q)).$$

We can also establish a connection between Rényi divergence of order α and $1-\alpha$ for $0 < \alpha < 1$.

$$\begin{aligned} D_{1-\alpha}(P\|Q) &= \frac{1}{-\alpha}\log\sum_{i=1}^{n}p_i^{1-\alpha}q_i^{\alpha}, \\ &= \frac{1-\alpha}{\alpha}\left(\frac{\alpha}{1-\alpha}\frac{1}{-\alpha}\log\sum_{i=1}^{n}q_i^{\alpha}p_i^{1-\alpha}\right), \\ &= \frac{1-\alpha}{\alpha}D_{\alpha}(Q\|P). \end{aligned}$$

Example. Let us have a probability distributions $Q = (\frac{4}{10}, \frac{6}{10})$ and $P = (p, 1-p)$ for some $p \in [0,1]$. On the left side of Figure 1.1, we plot $D_\alpha(P\|Q)$ as a function of p, for different values of α. Clearly when $p = \frac{4}{10}$, the divergence is zero for any α since both distributions are identical. Additionally, the divergence remains the same for any α, when $p = 0$ or $p = 1$. This follows from the fact that $D_\alpha(P\|Q) = \frac{1}{\alpha-1}\log q_1^{1-\alpha} = -\log q_1$ when $p = 1$ and $D_\alpha(P\|Q) = -\log q_2$ when $p = 0$.

On the right side of Figure 1.1, we plot $D_\alpha(P\|Q)$ as a function of q, where we now set $P = (\frac{4}{10}, \frac{6}{10})$ and $Q = (q, 1-q)$. Again, the divergence is always equal to zero, when the distributions are identical, e.g. when $q = \frac{4}{10}$. However, for $q = 0$ or $q = 1$, we obtain the expression $\frac{x}{0}$ for $x > 0$, which as defined in Definition 4, is set to ∞ for $\alpha \in (0,1) \cap (1,\infty)$. Thus, the divergence is also ∞. For $\alpha = 1$, the divergence is also infinite,, as follows from Definition 3. Interestingly, for $\alpha = 0$, from (1.2) we derive that the divergence is 0, which it is for any q.

For other values of α, the divergence varies, but a clear ordering emerges. That is, in the first example, the value of $D_\alpha(P\|Q)$ for $\alpha > \beta$ is greater than or equal to $D_\beta(P\|Q)$ for any $p \in [0,1]$, this is also true for the second example for any $p \in [0,1]$. Moreover, as shown by van Erven and Harremoës [2012], this holds in general, as Rényi divergence is non-decreasing in α.

Additionally, in the first example, we observe that for larger values of α the derivative is greater when p is close to $\frac{4}{10}$, whereas it is smaller when p is near 0 or 1. The opposite holds for smaller values of α. Conversely, for the second example, the derivative is high as we get near 0 or 1, actually it converges to

infinity for $\alpha > 0$. As shown in the figure, the rate of this convergence is slower for small values of α.

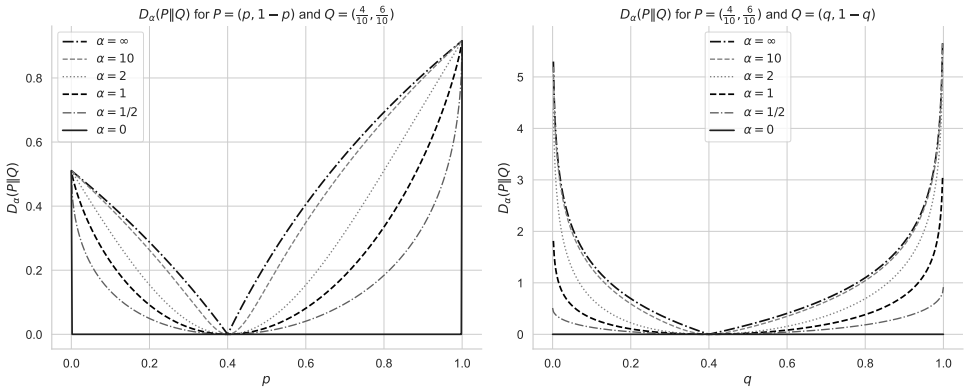

Figure 1.1: Example of Rényi divergence for a fixed distribution and another varying along the x-axis.

The final property we focus on is the lower semi-continuity.

Theorem 1. *Suppose we have a discrete sample space $\mathcal{Z} = \{z_1, z_2, z_3, \cdots\}$ and sigma-algebra $\mathcal{A}$ is the power set of $\mathcal{Z}$. Then, for any order $\alpha \in (0, \infty]$, the Rényi divergence is a lower semi-continuous function of the pair (P, Q) in the weak topology.*

Proof. Let $P_1, P_2, \cdots$ and $Q_1, Q_2, \cdots$ be sequences of discrete distributions that weakly converge to P and Q, respectively. We need to show

$$\liminf_{n\to\infty} D_\alpha(P_n\|Q_n) \geq D_\alpha(P\|Q).$$

Firstly, the weak convergence of discrete distribution P means that for every bounded continuous function h

$$\int h dP_n \to \int h dP.$$

As the sample set is discrete, we may set $h = \mathbb{1}\{z_i\}$ for any $i \in \mathbb{N}$, which is a bounded continuous function, and we obtain

$$P_n(z_i) = \int \mathbb{1}\{z_i\} dP_n \to \int \mathbb{1}\{z_i\} dP = P(z_i).$$

Now, any measurable set $A \in \mathcal{A}$ is just union of the individual elementary outcomes z_i and thus the probability on any set A is just the sum of the probabilities of the elementary outcomes. Using the convergence above we get

$$P_n(A) = \sum_{z_i \in A} P_n(z_i) \to \sum_{z_i \in A} P(z_i) = P(A),$$

and we proved that sequences $P_1, P_2, \cdots$ and $Q_1, Q_2, \cdots$ also converge pointwise to P and Q, respectively. Thus, also the sequence of the pairs (P_n, Q_n) converges pointwise to (P, Q).

Now, we can apply Fatou's lemma term-by-term on the sum

$$\liminf_{n\to\infty} \sum_i P_n(z_i)^\alpha Q_n(z_i)^{1-\alpha} \geq \sum_i \liminf_{n\to\infty} P_n(z_i)^\alpha Q_n(z_i)^{1-\alpha} \geq \sum_i P(z_i)^\alpha Q(z_i)^{1-\alpha}.$$

Taking the logarithm and scaling the by $\frac{1}{\alpha-1}$ preserves this inequality, thus yielding the lower semi-continuity of $D_\alpha(P\|Q)$.

□

1.2 Knowledge Distillation

Let us define a machine learning model as a function that maps input data to output predictions

$$f_\theta : \mathcal{X} \to \mathcal{Y}$$

where $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the output space and θ is a vector representing the set of parameters of the model. In our case, as input, the model receives images from $\mathcal{X}$, where each image is represented as a tensor in $\mathbb{R}^{h\times w\times c}$, where h and w denote the height and width in pixels, respectively, and c represents the number of channels, where for RGB images $c = 3$. We assume a classification task with n classes, i.e., $\mathcal{Y} = \mathbb{R}^n$, so the model outputs a vector of n real-valued scores, referred to as logits, given by

$$z = f_\theta(x)$$

for $x \in \mathcal{X}$. To convert these logits into probabilities, we use the softmax function, which is defined as

$$\sigma(s)_c = \frac{e^{s_c}}{\sum_{k=1}^n e^{s_k}}, \qquad c = 1, 2, \cdots, n, \tag{1.6}$$

where $s \in \mathcal{Y}$. Now let

$$q_c = \sigma(z)_c, \qquad c = 1, 2, \cdots, n,$$

which represents the probability that x belongs to class c. We denote the probability distribution produced by the model f_θ as $Q = (q_1, q_2, \cdots, q_n)$.

Additionally, a hyperparameter T, called temperature, is introduced to control the entropy of the output distribution. That is we set for $T > 0$

$$q_c^T = \sigma\left(\frac{z}{T}\right)_c, \qquad c = 1, 2, \cdots, n.$$

The produced probability distribution is $Q^T = \left(q_1^T, q_2^T, \cdots, q_n^T\right)$. The process in called temperature scaling and popular choices for T according to Cho and Hariharan [2019] are 3, 4 and 5. Clearly, $Q^1 = Q$.

Example. Following the example from the introduction, suppose a model f_θ that for given input yields logits for the classes "horse", "zebra", and "car", equal to 5.4, 0.2, and -1.3 respectively. In Figure 1.2, we see the logit values in a bar chart, along with the computed probabilities using the softmax function, both without and with temperature scaling, the latter corresponding to $T = 4$.

Without temperature scaling, the model is highly confident that the input belongs to the class horse (> 0.99), while the probabilities for the remaining classes are essentially zero. We observe that the effect of the temperature scaling is that the model is less confident about the true label while the order of the class probabilities in maintained.

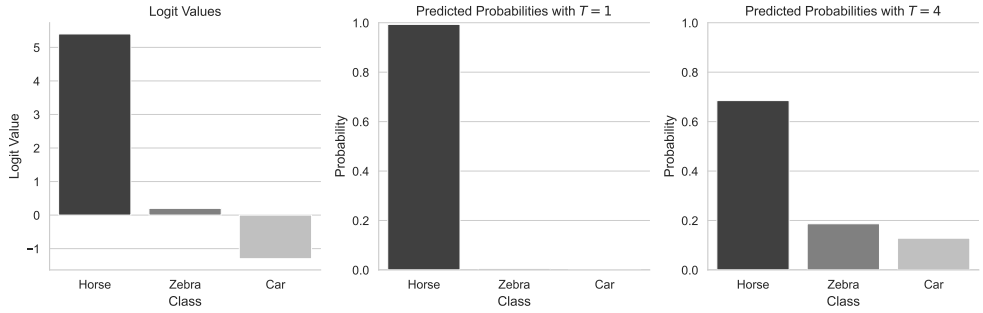

Figure 1.2: Example of temperature scaling.

Let $\mathcal{D}(x, y)$ denote a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = \mathbb{R}^n$, from which data points (x, y) are independently and identically distributed (i.i.d.) samples. Now, the cross-entropy loss function $\mathcal{L}_{\text{CE}}$ is defined by

$$\mathcal{L}_{\text{CE}}(\theta, x, y) = H(y\|Q) = -\sum_{j=1}^{n} y_j \log q_j, \tag{1.7}$$

where Q is the probability distribution obtained by applying the softmax function to the model's output $f(x)$, and y_i is the one-hot ground-truth label.

Training model f_θ is the process of finding the parameters θ that minimize the expected loss with respect to the distribution $\mathcal{D}$. That is, we want find θ^* such that

$$\theta^* = \arg\min_{\theta} \mathsf{E}_{(x,y)\sim\mathcal{D}} \mathcal{L}_{\text{CE}}(\theta).$$

Although KL divergence provides a more intuitive measure of the difference between two distributions, being zero when the distributions are equal, unlike cross-entropy, we did not use it in Equation 1.7. As shown in Equation 1.1, $H(P)$ does not depend on Q. Thus, the derivative of $D_{\text{KL}}(P\|Q)$ with respect to Q is equal to the derivative of $H(P\|Q)$. Moreover, since cross-entropy is computationally simpler, especially when P represents one-hot labels, it is often preferred over KL divergence in machine learning applications.

Remark. In the context of knowledge distillation, the training process defined in Equation 1.7 is referred to as vanilla training. This serves as a benchmark against which we compare the results of knowledge distillation.

Definition 5. *Knowledge distillation is a model compression technique where a smaller student model f_θ is trained to mimic a larger teacher model f_t, which has been already pre-trained. Let $\mathcal{D}(x, y)$ denote a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = \mathbb{R}^n$, and $(x, y) \sim \mathcal{D}$. The student model outputs logits $z = f_\theta(x)$, which are converted into probability distribution $Q^T = (q_1^T, \cdots, q_n^T)$*

using softmax function introduced in Equation 1.6,

$$q_c^T = \sigma\left(\frac{z}{T}\right)_c = \frac{e^{\frac{z_c}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}}, \qquad c = 1, 2, \cdots, n.$$

Similarly, the teacher model outputs logits $v = f_t(x)$, which are converted into probability distribution $P^T = (p_1^T, \cdots, p_n^T)$ using softmax function,

$$p_c^T = \sigma\left(\frac{v}{T}\right)_c = \frac{e^{\frac{v_c}{T}}}{\sum_{k=1}^n e^{\frac{v_k}{T}}}, \qquad c = 1, 2, \cdots, n.$$

The goal of the training process is to optimizes the loss function of the form

$$\mathcal{L}(\theta, x, y) = (1-\beta)\mathcal{L}_{CE}(\theta, x, y) + \beta\mathcal{L}_{KL}(\theta, x, y), \tag{1.8}$$

where $\mathcal{L}_{CE}(\theta, x, y)$ is the standard cross-entropy loss with ground truth labels

$$\begin{aligned} \mathcal{L}_{CE}(\theta, x, y) &= H(y_i \| Q), \\ &= -\sum_{j=1}^n y_{i,j} \log q_j, \end{aligned} \tag{1.9}$$

and $\mathcal{L}_{KL}(\theta, x, y)$ is the Kullback-Leibler divergence loss with teacher's predictions

$$\begin{aligned} \mathcal{L}_{KL}(\theta, x, y) &= T^2 D_{KL}(P^T \| Q^T), \\ &= T^2 \sum_{j=1}^n p_j^T \log \frac{p_j^T}{q_j^T}, \end{aligned} \tag{1.10}$$

T and β are hyperparameters.

The hyperparameter T in Definition 5 denotes temperature. During training, we apply temperature scaling to both the teacher and the student in Equation 1.10. By increasing T, we soften the probabilities, thus retaining inter-class similarities by driving the predictions away from 0 and 1. The second hyperparameter, β, controls the balance between the cross-entropy loss and Kullback-Leibler divergence loss. A common choice for β is 0.9 (see Cho and Hariharan [2019]).

We observe that, unlike in Equation 1.9, which is simply the cross-entropy, in Equation 1.10, the loss function also includes the term T^2. To understand the origin of this term we first calculate the derivatives of KL divergence $D_{\mathrm{KL}}(P^T \| Q^T)$ with respect to the logits of Q^T. We compute the following

$$\begin{aligned} \frac{\partial D_{\mathrm{KL}}(P^T \| Q^T)}{\partial z_j} = \frac{\partial H(P^T \| Q^T)}{\partial z_j} &= -\frac{\partial}{\partial z_j} \sum_{i=1}^n p_i^T \log \frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}}, \\ &= \left(\sum_{i=1}^n p_i^T\right) \frac{\partial}{\partial z_j} \log \sum_{k=1}^n e^{\frac{z_k}{T}} - \frac{\partial}{\partial z_j} \sum_{i=1}^n p_i^T \frac{z_i}{T}, \\ &= \frac{1}{T} \frac{e^{z_j}}{\sum_{k=1}^n e^{z_k}} - \frac{p_j^T}{T}, \\ &= \frac{1}{T}(q_j^T - p_j^T). \end{aligned}$$

Thus, the derivative of both KL divergence and cross-entropy is given by

$$\frac{\partial D_{\mathrm{KL}}(P^T \| Q^T)}{\partial z_j} = \frac{\partial H(P^T \| Q^T)}{\partial z_j} = \frac{1}{T}(q_j^T - p_j^T). \tag{1.11}$$

Now, similarly to Hinton et al. [2015], we assume centered logits $\sum_{k=1}^n z_k = \sum_{k=1}^n v_k = 0$. Then we have

$$\frac{\partial H(P^T \| Q^T)}{\partial z_j} = \frac{1}{T}(q_j^T - p_j^T) = \frac{1}{T}\left(\frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}} - \frac{e^{\frac{v_j}{T}}}{\sum_{k=1}^n e^{\frac{v_k}{T}}}\right). \tag{1.12}$$

Now, we approximate the exponential function using a Taylor polynomial of degree one and we obtain

$$\frac{\partial H(P^T \| Q^T)}{\partial z_j} \approx \frac{1}{T}\left(\frac{1 + \frac{z_j}{T}}{n + \sum_{k=1}^n \frac{z_k}{T}} - \frac{1 + \frac{v_j}{T}}{n + \sum_{k=1}^n \frac{v_k}{T}}\right) = \frac{1}{nT^2}(z_j - v_j). \tag{1.13}$$

Thus, the loss gradient decreases proportionally to $\frac{1}{T^2}$ as the temperature T increases. By incorporating the term T^2 into Equation 1.10, we ensure that the relative contribution of $\mathcal{L}_{\mathrm{CE}}(\theta, x, y)$ and $\mathcal{L}_{\mathrm{KL}}(\theta, x, y)$ remains approximately the same.

For lower temperature, where the approximation by the Taylor polynomial is very inaccurate, Hinton et al. [2015] states that the distillation pays less attention to matching logits much more negative than average. This is advantageous, as they may be significantly noisier, given that the teacher model is not penalized for them during training. On the other hand, they might convey useful information about the knowledge acquired by the teacher. Based on empirical evidence, the authors claim that ignoring large negative logits has a positive effect, as intermediate temperatures yield the best results.

Now, we replace the KL divergence loss in Definition 5 by a general Rényi divergence of order α. Thus, the loss function (1.8) is replaced by

$$\mathcal{L}(\theta, x, y) = (1 - \beta)\mathcal{L}_{\mathrm{CE}}(\theta, x, y) + \beta\mathcal{L}_{\alpha}(\theta, x, y), \tag{1.14}$$

where

$$\begin{aligned} \mathcal{L}_{\alpha}(\theta, x, y) &= \frac{T^2}{\alpha} D_{\alpha}(P^T \| Q^T), \\ &= \frac{T^2}{\alpha} \sum_{j=1}^n \frac{1}{\alpha - 1} \log (p_j^T)^{\alpha} (q_j^T)^{1-\alpha}. \end{aligned} \tag{1.15}$$

Similarly to above for (1.10) we elaborate on the normalizing factor in Equation 1.15 which is now equal to $\frac{T^2}{\alpha}$. Firstly, we compute the derivatives of $D_{\alpha}(P^T \| Q^T)$ with respect to the logits of Q^T. For simplicity, we omit T in P^T, Q^T, p^T and q^T. We have

$$\frac{\partial D_{\alpha}(P \| Q)}{\partial z_j} = \frac{\partial}{\partial z_j} \frac{1}{\alpha - 1} \log \sum_{i=1}^n p_i^{\alpha} \left(\frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}}\right)^{1-\alpha}.$$

Now denote $Z = \sum_{i=1}^n p_i^{\alpha} \left(\frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}}\right)^{1-\alpha}$. By the chain rule we have

$$\frac{\partial D_\alpha(P\|Q)}{\partial z_j} = \frac{\partial}{\partial z_j}\frac{1}{\alpha-1}\log Z = \frac{1}{\alpha-1}\frac{1}{Z}\frac{\partial Z}{\partial z_j}.$$

Now we need to calculate $\frac{\partial Z}{\partial z_j}$.

$$\begin{aligned}
\frac{\partial Z}{\partial z_j} =& \frac{\partial}{\partial z_j}\sum_{i=1}^{n} p_i^\alpha \left(\frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right)^{1-\alpha}, \\
=& \frac{\partial}{\partial z_j} p_j^\alpha \left(\frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right)^{1-\alpha} + \frac{\partial}{\partial z_j}\sum_{i\neq j}^{n} p_i^\alpha \left(\frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right)^{1-\alpha}, \\
=& p_j^\alpha(1-\alpha)\left(\frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right)^{-\alpha} \frac{\frac{1}{T}\sum_{k=1}^{n} e^{\frac{z_k}{T}} e^{\frac{z_j}{T}} - \frac{1}{T}e^{\frac{z_j}{T}}e^{\frac{z_j}{T}}}{(\sum_{k=1}^{n} e^{\frac{z_k}{T}})^2} \\
& - \sum_{i\neq j}^{n} p_i^\alpha \left(\frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right)^{-\alpha} (1-\alpha)\frac{e^{\frac{z_i}{T}}}{(\sum_{k=1}^{n} e^{\frac{z_k}{T}})^2} e^{\frac{z_j}{T}}\frac{1}{T}, \\
=& \frac{1-\alpha}{T}\left[p_j^\alpha \left(\frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right)^{1-\alpha} \frac{\sum_{k=1}^{n} e^{\frac{z_k}{T}} - e^{\frac{z_j}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right. \\
& \left. - \sum_{i\neq j}^{n} p_i^\alpha \left(\frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right)^{1-\alpha} \frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^{n} e^{\frac{z_k}{T}}}\right], \\
=& \frac{1-\alpha}{T}\left[p_j^\alpha q_j^{1-\alpha}(1-q_j) - \sum_{i\neq j}^{n} p_i^\alpha q_i^{1-\alpha} q_j\right], \\
=& \frac{1-\alpha}{T}\left[p_j^\alpha q_j^{1-\alpha} - q_j\sum_{i=1}^{n} p_i^\alpha q_i^{1-\alpha}\right], \\
=& \frac{1-\alpha}{T}(p_j^\alpha q_j^{1-\alpha} - q_j Z).
\end{aligned}$$

Now, inserting $\frac{\partial Z}{\partial z_j}$ into the original equation, we obtain

$$\frac{\partial D_\alpha(P\|Q)}{\partial z_j} = \frac{1}{\alpha-1}\frac{\frac{1-\alpha}{T}(p_j^\alpha q_j^{1-\alpha} - q_j Z)}{Z} = \frac{q_j Z - p_j^\alpha q_j^{1-\alpha}}{TZ}.$$

We can also substitute the original expression for Z and simplify the result to derive

$$\frac{\partial D_\alpha(P\|Q)}{\partial z_j} = \frac{1}{T}\left(q_j - \frac{p_j^\alpha q_j^{1-\alpha}}{\sum_{i=1}^{n} p_i^\alpha q_i^{1-\alpha}}\right), \tag{1.16}$$

where for $\alpha = 1$, we arrive at the same result as for KL divergence.

If the distribution P represents one-hot labels, the derivative simplifies to the same form as in Equation 1.12. This holds for any choice of α, which is why we do not modify the $\mathcal{L}_{\text{CE}}$ term in knowledge distillation or vanilla training.

From the result in Equation 1.16, we derive an analogous expression to Equation 1.13, using previously established notation and assumption of centered logits. Moreover, we approximate the exponential function using a Taylor polynomial of degree one. We derive

$$
\begin{aligned}
\frac{\partial D_\alpha(P\|Q)}{\partial z_j} =& \frac{1}{T}\left[\frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}} - \left(\frac{e^{\frac{v_j}{T}}}{\sum_{k=1}^n e^{\frac{v_k}{T}}}\right)^\alpha \left(\frac{e^{\frac{z_j}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}}\right)^{1-\alpha}\right. \\
&\left.\left(\sum_{i=1}^n \left(\frac{e^{\frac{v_i}{T}}}{\sum_{k=1}^n e^{\frac{v_k}{T}}}\right)^\alpha \left(\frac{e^{\frac{z_i}{T}}}{\sum_{k=1}^n e^{\frac{z_k}{T}}}\right)^{1-\alpha}\right)^{-1}\right], \\
\approx& \frac{1}{T}\left[\frac{1+\frac{z_j}{T}}{n+\sum_{k=1}^n \frac{z_k}{T}} - \frac{1+\frac{\alpha v_j}{T}}{\left(n+\sum_{k=1}^n \frac{v_k}{T}\right)^\alpha}\frac{1+\frac{(1-\alpha)z_j}{T}}{\left(n+\sum_{k=1}^n \frac{z_k}{T}\right)^{1-\alpha}}\right. \\
&\left.\left(\sum_{i=1}^n \frac{1+\frac{\alpha v_i}{T}}{\left(n+\sum_{k=1}^n \frac{v_k}{T}\right)^\alpha}\frac{1+\frac{(1-\alpha)z_i}{T}}{\left(n+\sum_{k=1}^n \frac{z_k}{T}\right)^{1-\alpha}}\right)^{-1}\right], \\
=& \frac{1}{T}\left[\frac{1+\frac{z_j}{T}}{n} - \frac{1+\frac{\alpha v_j}{T}+\frac{(1-\alpha)z_j}{T}+\frac{\alpha(1-\alpha)v_j z_j}{T^2}}{n}\right. \\
&\left.\left(\frac{1}{n}\sum_{i=1}^n 1+\frac{\alpha v_i}{T}+\frac{(1-\alpha)z_i}{T}+\frac{\alpha(1-\alpha)v_i z_i}{T^2}\right)^{-1}\right].
\end{aligned}
$$

When using the approximation of the exponential function by the first Taylor polynomial, we assume that the higher-order terms are negligible. Particularly, this means that $\alpha^2 v_j^2 = o(T^2)$ and $(1-\alpha)^2 z_j^2 = o(T^2)$. From this we get αv_j and $(1-\alpha)z_j$ are $o(T)$. Thus, their product is $o(T^2)$, which means that $\frac{\alpha(1-\alpha)v_j z_j}{T^2} \approx 0$.

This, together with the previously mentioned assumption of zero-meaned logits, allows us to further simplify the formula.

$$
\frac{\partial D_\alpha(P\|Q)}{\partial z_j} \approx \frac{1}{T}\left[\frac{1+\frac{z_j}{T}}{n} - \frac{1+\frac{\alpha v_j}{T}+\frac{(1-\alpha)z_j}{T}}{n}\left(\frac{n}{n}\right)^{-1}\right] = \frac{\alpha}{nT^2}(z_j - v_j).
$$

This formula is similar to that of KL divergence (1.13), except that it is multiplied by α.

2. Stochastic Gradient Descent and Residual Neural Network

In the previous section, we defined the loss function for knowledge distillation incorporating Rényi divergence. In this chapter, we discuss how to minimize this function using the stochastic gradient descent algorithm. Additionally, we describe the Residual Neural Network architecture, which will be used as a model both for the teacher and the student model in the sequel.

2.1 Stochastic Gradient Descent for Knowledge Distillation

Denote $\mathcal{D}(x, y)$ as a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = \mathbb{R}^n$, and $(x, y) \sim \mathcal{D}$. Let f_θ be a student model as defined in Definition 5, with a parameter vector θ. Also denote f_t a pre-trained model, referred to as a teacher. During training the goal is to minimize the loss function with respect to the parameters θ. Our loss function as defined in Equation 1.14 is given by

$$\mathcal{L}(\theta, x, y) = (1 - \beta)\mathcal{L}_{\mathrm{CE}}(\theta, x, y) + \beta\mathcal{L}_\alpha(\theta, x, y), \tag{2.1}$$

where $\mathcal{L}_{\mathrm{CE}}(\theta, x, y)$ is the cross-entropy loss, $\mathcal{L}_\alpha(\theta, x, y)$ is Rényi divergence loss, $\alpha \geq 0$ and $\beta \in (0, 1)$.

We define the expected risk of a model f_θ given a loss function $\mathcal{L}$ as

$$E(f_\theta) = \mathsf{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}(\theta, x, y)\right], \tag{2.2}$$

where the expectation is take with respect to the joint distribution of (x, y).

The expected risk measures the generalization performance of the model f_θ. Unfortunately, the distribution of (x, y) is unknown, thus we approximate the expected risk by the empirical risk for over a given dataset $\mathcal{S} = \{(x_i, y_i) \sim \mathcal{D}, \text{independently for } i = 1, \ldots, N\}$, which is defined as

$$E_N(f_\theta) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\theta, x_i, y_i).$$

The gradient descent (GD) algorithm, adopted by Rumelhart et al. [1986] for training neural networks, aims to minimize the empirical risk $E_N(f_\theta)$. After initialization, in each iteration, called an epoch, the parameters θ are updated using the gradient of the loss function as follows

$$\theta_{t+1} = \theta_t - \gamma\frac{1}{N}\sum_{i=1}^{N}\nabla_\theta\mathcal{L}(\theta_t, x_i, y_i),$$

where θ_t are the parameters of the model after t iterations of the gradient descent algorithm, γ is the learning rate and $(x_i, y_i) \in \mathcal{S}$. The number of iterations and learning rate γ are hyperparameters. This algorithm is sometimes called the total gradient algorithm.

A simplification of the previous algorithm is the stochastic gradient descent (SGD) algorithm. In each iteration, the parameters θ are updated as follows

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta \mathcal{L}(\theta_t, x_j, y_j),$$

where $(x_j, y_j) \in \mathcal{S}$ is a single randomly chosen datapoint from the training dataset.

As noted by Bottou [2010], SGD directly optimizes the expected risk (2.2) since the datapoints are randomly drawn from the ground truth distribution.

According to Bottou [1991], there are few advantages of using SGD over GD. Such as, in datasets with redundancy, only a small subset of datapoints is needed to obtain a good estimate of the gradient, making SGD more efficient. Also, while gradient descent may converge to a local minimum from which it cannot escape, the random effect in SGD often prevents such behavior.

On the other hand, the main drawback of stochastic gradient descent is the high variance of the estimator of the expected risk, as it relies on a singular sample per iteration. To retain the advantages while reducing variance, the mini-batch stochastic gradient descent algorithm can be introduced. In each iteration, a random subset (mini-batch) of $B < N$ data points is sampled from the training dataset $\mathcal{S}$. The algorithm is defined as follows

$$\theta_{t+1} = \theta_t - \gamma \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\theta_t, x_i, y_i),$$

where $\mathcal{B}_t = \{(x_i, y_i)_{i=1}^B\}$ denotes mini-batch sampled at iteration t.

Additionally, we introduce weight decay λ regularization term to discourage large values of θ, along with Nesterov momentum μ based on the formula from Sutskever et al. [2013]. Both λ and μ are hyperparameters. The resulting algorithm is as follows

$$\begin{aligned} b_{t+1} &= \mu b_t + \left(\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\theta_t, x, y) + \lambda \theta_t \right), \\ \theta_{t+1} &= \theta_t - \gamma \left(\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(\theta_t, x, y) + \lambda \theta_t + \mu b_{t+1} \right), \end{aligned}$$

where $\mathcal{B}_t = \{(x_i, y_i)_{i=1}^B\}$ denotes mini-batch sampled at iteration t.

In practice, an adjustment to this method is used, where in each epoch, the dataset is randomly shuffled and divided into mini-batches of size $B < N$, and each mini-batch is processed sequentially — so that every data point is used once per epoch.

Let $z = f_{\theta_t}(x)$ be the computed logits of the student model. The gradient of the loss function is calculated using backpropagation, that is

$$\nabla_\theta \mathcal{L}(\theta_t, x, y) = \frac{\partial \mathcal{L}(\theta_t, x, y)}{\partial \theta} = \frac{\partial \mathcal{L}(\theta_t, x, y)}{\partial z} \frac{\partial z}{\partial \theta},$$

where z is the vector of the logits that the model outputs. From Equations 1.11, 1.15, 1.16 and 2.1 we get

$$\begin{aligned}\frac{\partial \mathcal{L}(\theta_t, x, y)}{\partial z} &= (1-\beta)\frac{\partial \mathcal{L}_{\text{CE}}(\theta_t, x, y)}{\partial z} + \beta\frac{\partial \mathcal{L}_{\alpha}(\theta_t, x, y)}{\partial z}, \\ &= (1-\beta)(Q-y) + \beta\frac{T}{\alpha}\left(Q^T - \frac{(P^T)^{\alpha} \odot (Q^T)^{1-\alpha}}{\langle (P^T)^{\alpha}, (Q^T)^{1-\alpha} \rangle}\right),\end{aligned}$$

where P^T, Q^T and Q as in Definition 5. Note that in the cross-entropy part of the equation the temperature scaling is not used. Let us also denote $\odot$ as the element-wise product, $\langle \cdot, \cdot \rangle$ as the scalar product, and $(\cdot)^a$ as element-wise exponentiation. Thus, the gradient used in SGD for knowledge distillation using Rényi divergence loss is given by

$$\nabla_{\theta}\mathcal{L}(\theta_t, x, y) = \left[(1-\beta)(Q-y) + \beta\frac{T}{\alpha}\left(Q^T - \frac{(P^T)^{\alpha} \odot (Q^T)^{1-\alpha}}{\langle (P^T)^{\alpha}, (Q^T)^{1-\alpha} \rangle}\right)\right]\frac{\partial z}{\partial \theta}.$$

What remains to be calculated is $\frac{\partial z}{\partial \theta}$, which depends on the architecture of the student model.

2.2 Residual Neural Network

When using knowledge distillation, we need to define the architectures of both the teacher and student models. In this thesis, we chose to use a Residual Neural Network (ResNet) introduced in He et al. [2016]. This architecture is prominent in computer vision, as it addresses the problems of degradation and vanishing/exploding gradients in deep neural networks.

The architecture of ResNet consists of fully connected layers, convolutional layers, and pooling layers. To understand the fully connected layer, we first define a neuron. Neuron is a function that maps $x \in \mathbb{R}^k$ onto $z \in \mathbb{R}$ as

$$z = h(\sum_{i=1}^{k} w_i x_i + b),$$

where w_i represents the weight between the i-th input and the neuron, b is the bias, and h is the activation function. Commonly used activation functions include the identity function, sigmoid, hyperbolic tangent, and the rectified linear unit (ReLU), defined as

$$\text{ReLU}(x) = \max(0, x),$$

which is the activation function used in ResNet. ReLU is primarily used for its simplicity and properties of its derivative (see Agarap [2018]).

A fully connected layer in a neural network consists of multiple neurons with common activation function operating in parallel, each computing an output z_j based on its own weights and biases. The output of a layer is a vector of all individual neuron outputs, i.e. $z = (z_1, \cdots, z_l)$, where l represents the number of neurons in a layer.

To create a multi-layer neural network, the output of one layer is fed as the input to the next. The size of each layer may vary, and we denote the total number of layers by L.

Another type of layer is the convolutional layer. Here, the input to the layer is $x \in \mathbb{R}^{h \times w \times c}$, representing an image, where h denotes height, w width and c number of channels. For fixed channel k, the output is also an image y given by

$$y_{i,j,k} = \sum_{m=-H}^{H} \sum_{n=-W}^{W} x_{i+m,j+n,k} \cdot K_{m,n}, \tag{2.3}$$

where $i = 1 + H, \cdots, h - H$, $j = 1 + W, \cdots, w - W$, $K \in \mathbb{R}^{k_h \times k_q}$ is so-called kernel, $k_h = 2H + 1$ and $k_w = 2W + 1$. We mostly consider only square kernels, i.e. $k_h = k_w$, usually $k_h = k_w \in \{1, 3, 5, 7\}$.

As we see, the output has a different dimension $y \in \mathbb{R}^{(h-2H) \times (w-2W) \times c}$, compared to the input x. This is often undesirable, so we use a method called padding. Using this technique, y is computed as in 2.3 for all $i = 1, \cdots, h$ and $j = 1, \cdots, w$, while defining $x_{u,v,k} = 0$ for any $u < 1$, $u > h$, $v < 1$ or $v > w$.

It is common that there are multiple different kernels in one convolutional layer, each for a different channel, or even more. Thus, the output of a convolutional layer is a collection of images.

The advantage of convolution is that it focuses on local regions of an image, allowing it to detect patterns such as object edges. Additionally, convolution can recognize the same pattern regardless of its location in the image.

We also introduce stride, the output of a layer is computed as

$$y_{i,j,k} = \sum_{m=-H}^{H} \sum_{n=-W}^{W} x_{s \cdot i+m,s \cdot j+n,k} \cdot K_{m,n},$$

where $s \in \mathbb{N}$ is a stride. If $s = 2$, the output is half the height and width of the input. Thus, $y \in \mathbb{R}^{\frac{h}{2} \times \frac{w}{2} \times c}$.

Another type of layer is the pooling layer, which is similar to a convolutional layer with stride, typically set to 2. However, instead of applying a kernel, it applies a non-linear function to the local region. If the function returns the maximum value, it is called max pooling (MaxPool), if it returns the mean, it is called average pooling (AvgPool). There is also a specific type called global max pooling (GMP), where we take the maximum value over the entire image for each channel. As a result, the output is a vector $\mathbb{R}^c$.

A unique feature of a Residual Neural Network is the residual block. This residual block is composed of a function $\mathcal{F}$ and so-called shortcut, represented by an identity function. The function $\mathcal{F}$ represents multiple layers of the neural network and it the part of residual mapping to be learned. The number of layers in $\mathcal{F}$ may vary, but the authors used two layers. The output of this building block y then follows

$$y = \mathrm{ReLU}(\mathcal{F}(x) + x), \tag{2.4}$$

where x is the input. This process is depicted in Figure 2.1. If y and x are of a different dimension, we perform linear projection on x, thus, we get

$$y = \mathrm{ReLU}(\mathcal{F}(x) + Wx), \tag{2.5}$$

where W is a projection matrix of appropriate dimension.

The identity mapping helps the gradient propagate more effectively through the neural network, even in deep architectures. It also provides the network with

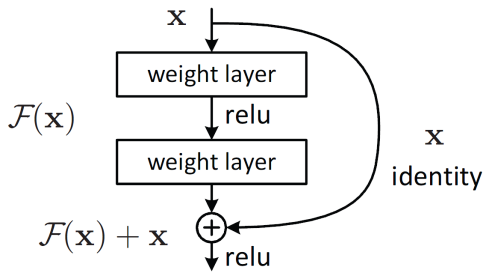

Figure 2.1: Residual block (He et al. [2016]).

the ability to skip the transformation $\mathcal{F}(x)$ entirely, in cases where the identity mapping is optimal. This also allows the network to learn residuals as small adjustments to the identity function, rather than requiring the network to learn a complete transformation from scratch.

Now, we can define the plain network as described in He et al. [2016], which serves as the basis for ResNet. The plain network was inspired by VGG nets, which were the state-of-the-art architecture at the time of publication (see Simonyan and Zisserman [2014]). The network starts with a convolutional layer with kernel size 7×7, continuing with many convolutional layers with kernel size 3×3. Two design rules are followed. When performing convolution with a stride of 1, the number of kernels matches the number of input channels. However, when using a stride of 2, the number of kernels doubles to preserve the complexity. The network ends with a global average pooling layer and a fully-connected layer. The plain network and type of VGG net (VGG-19) can be seen in Figure 2.2.

A Residual Neural Network is constructed from the plain network by introducing shortcuts, every two layers, as shown in Figure 2.2, where solid lines represent the use of Equation 2.4, and dotted lines represent the use of Equation 2.5.

There are different variants of ResNet models, distinguished by the number of layers, which affect their capacity and performance. The most commonly used variants are ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152, where the number indicates the total number of layers.

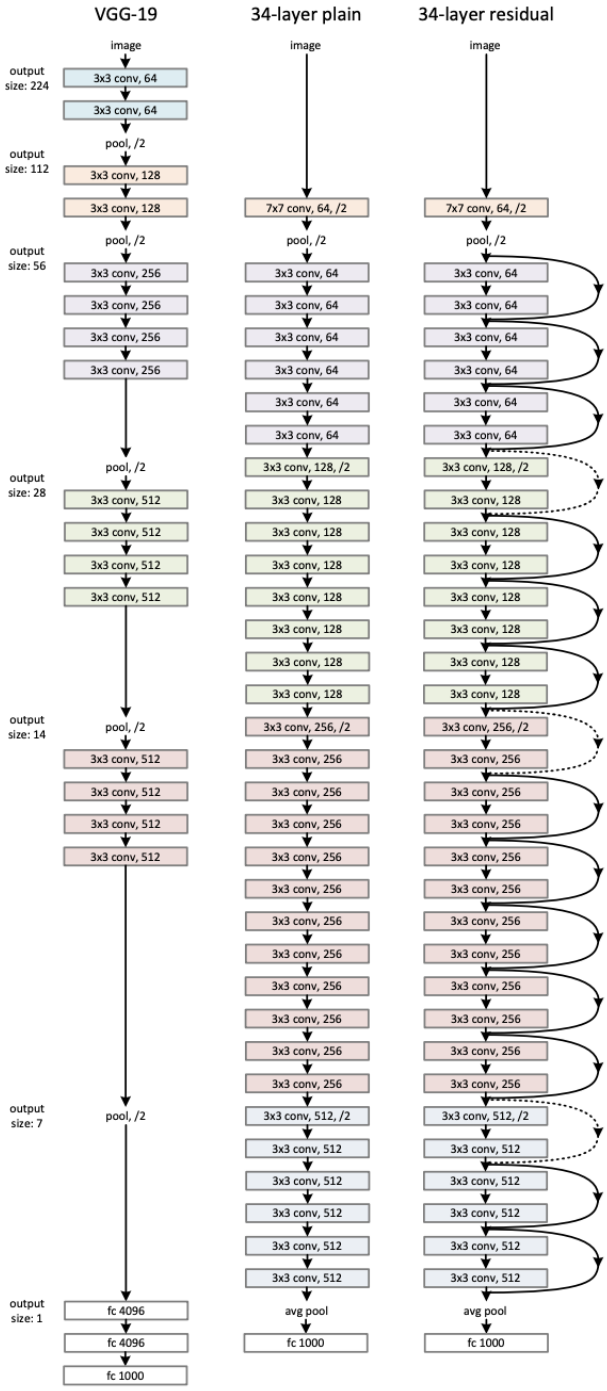

Figure 2.2: Example of network architecture (He et al. [2016]). Left: VGG-19 model as a reference. Middle: a plain network with 34 layers. Right: residual network with 34 layers (ResNet34).

Conclusion

Bibliography

Abien Fred Agarap. Deep learning using rectified linear units (relu). *ArXiv*, abs/1803.08375, 2018. URL https://api.semanticscholar.org/CorpusID:4090379.

Léon Bottou. Stochastic gradient learning in neural networks. 1991. URL http://leon.bottou.org/papers/bottou-91c.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. *Proc. of COMPSTAT*, 09 2010. doi: 10.1007/978-3-7908-2604-3_16.

JH Cheng, C Zheng, R Yamada, and D Okada. Visualization of the landscape of the read alignment shape of atac-seq data using hellinger distance metric. *Genes & Cells*, 29(1):5–16, January 2024. doi: 10.1111/gtc.13082. Epub 2023 Nov 21.

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. *CoRR*, abs/1910.01348, 2019. URL http://arxiv.org/abs/1910.01348.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

Alfréd Rényi. On measures of entropy and information. 1961. URL https://api.semanticscholar.org/CorpusID:123056571.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948. URL http://plan9.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL https://api.semanticscholar.org/CorpusID:14124313.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. page III–1139–III–1147, 2013.

Tim van Erven and Peter Harremoës. Rényi divergence and kullback-leibler divergence. *CoRR*, abs/1206.2459, 2012. URL http://arxiv.org/abs/1206.2459.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment