h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Darmstadt University of Applied Sciences

– Faculty of Computer Science –

NeutrEx-Lite: Efficient Expression Neutrality Estimation For Utility Prediction

by

Thanh Tung Linh Nguyen

Matriculation number: 1113587

First Examiner	:	Prof. Dr. Christoph Busch
Second Examiner	:	Marcel Grimmer

DECLARATION

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 15. April 2024

Thanh Tung Linh Nguyen

ZUSAMMENFASSUNG

Gesichtserkennungssysteme haben viele praktische Anwendungen, wie z. B. die Smartphone-Authentifizierung, die Grenzkontrolle an Flughäfen oder die Anwesenheitskontrolle in Klassenräumen oder Einrichtungen. Für diese Anwendungsfälle ist es wichtig, dass das System eine hohe Erkennungsgenauigkeit erzielt und gleichzeitig in der Lage ist, eine große Anzahl an Transaktionen zu verarbeiten. Um sicherzustellen, dass Bilder von geringer Qualität die biometrische Erkennungsleistung nicht beeinträchtigen, ist eine Methode zur Quantifizierung der Qualität von Gesichtsbildern erforderlich. Der aktuelle Komitee-Entwurf von der internationalen Norm ISO/IEC 29794-5 führt das Konzept der Komponentenqualität ein, das die Qualität eines Gesichtsbilder in Bezug auf ein individuelles Qualitätselement quantifitiv ausdrückt. In dieser Arbeit untersuchen wir NeutrEx - ein Komponentenqualitätsmaß, das die Ausdrucksneutralität von Gesichtsbildern im Kontext von ISO/IEC 29794-5 quantifiziert. Wir optimieren das NeutrEx-Modell, um eine bessere Effizienz in Bezug auf die Anzahl der Parameter, den Speicherplatz und die Inferenzzeit zu erreichen. Das von uns vorgeschlagene Modell NeutrEx-lite soll eine effizientere Version von NeutrEx sein, so dass es für reale Anwendungen besser geeignet ist, aber dennoch eine vergleichbare Performance zum ursprünglichen NeutrEx-Modell erzielt. Um dies zu erreichen, erforschen wir etablierte Methoden auf dem Gebiet der Optimierung neuronaler Netze, wie z.B. Pruning, Quantization und Knowledge Distillation, und wenden sie auf NeutrEx an. Der wissenschaftliche Beitrag dieser Arbeit kann wie folgt zusammengefasst werden:

- Analyse von NeutrEx im Hinblick auf die zugrunde liegende Architektur, um mögliche Optimierungspunkte zu identifizieren.
- Literatur-Recherche nach möglichen Optimierungstechniken, um die Effizienz von NeutrEx zu erhöhen.
- Implementierungen solcher Techniken und ein Benchmarking der Ergebnisse mit dem ursprünglichen NeutrEx-Modell.
- Präsentation und Diskussion der Ergebnisse mit anschließender Ausführung zukünftiger Forschungsrichtungen in Bezug auf NeutrEx und NeutrEx-lite.

ABSTRACT

Face recognition systems have many useful applications in real life scenario, such as authentication for personal devices, border control at airports or attendance control for classrooms or institution. For these use cases, it is important for the system to achieve high recognition performance while still be able to process a large amount of transactions. To ensure that low quality images do not negatively contribute to the biometric performance, a method to quantify the quality of images taken from biometric samples is highly desirable. The current draft international standard of ISO/IEC 29794-5 introduces the concept of component quality, which quantitatively expresses the quality of a given biometric sample. In this work, we look into NeutrEx - a recently proposed quality measure which quantifies the expression neutrality of facial images in the context of ISO/IEC 29794-5. Additionally, we optimize the NeutrEx model to achieve better efficiency with regard to number of parameters, storage space, and inference time. Our proposed model NeutrEx-lite is intended to be a more streamlined version of NeutrEx, such that it becomes more suitable for real life applications, while still maintain respectable performance relative to the original one. In order to achieve this, we research well-known methods in the field of neural network optimization; such as pruning, quantization and knowledge distillation and apply them to NeutrEx. Our contributions are as follow:

- Analysis of NeutrEx with regard to the underlying architecture to identify possible optimization spots.
- Research into possible optimization techniques in the literature that can help us increase the efficiency of NeutrEx.
- Concrete implementations of such techniques and benchmark the results against the original NeutrEx.
- Observations and discussion of final result and possible future works with regard to NeutrEx and NeutrEx-lite.

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

The image capturing process of biometric characteristics is crucial to the functions of any biometric systems, as it can have a huge impact on the effectiveness of the system [AFFOG12; OR+22]. As such, it is important to ensure that the representation of captured biometric samples used in the system is of high quality. In biometric research, *utility* refers to how a biometric sample contributes to the overall performance of a biometric system. The current draft of ISO/IEC 29794-5 [ISO23b] extends utility into *unified quality* and *component quality*. Unified quality assess the overall image quality, while component quality shows how individual quality component influences the recognition performance.

Face recognition (FR) systems are one of the most widely used biometric systems. Examples of this can be found at large scale, such as the Entry/Exit System [DA18] of the European Union; or for personal usage, as implemented in face authentication components on modern computational devices [LLC22; Inc22]. Recent studies [Peñ+21; Dam+18] show that recognizing facial images which significantly deviate from neutral expression is a challenging task for current approaches. Among the quality components listed in the current draft of ISO/IEC 29794-5, *expression neutrality* is an important factor for the recognition capability of a face recognition system. Not only that, expression neutrality is proven to be difficult to assess, as pointed out by [SL03; MV16]. Grimmer et al. [Gri+23] propose a novel method to measure expression neutrality which is also compatible with the current committee draft of ISO/IEC 29794-5 [Iso]. This method - named **NeutrEx** by the authors - shows promising results compared to other approaches which follow the *Open Source Face Image Quality (OFIQ)* framework[1]. Furthermore, as the method is based on recent advancements in 3D facial reconstruction, it also opens the possibility to transform the classification results into actionable feedback for users, which can be helpful in real life application. However, as noted by the authors in their publication, NeutrEx is not without drawbacks, most importantly the increase in computational power by the proposed algorithm: The EMOCA [DBB22] and DECA [Fen+21] encoders incorporated in the architecture require up to 50 million input parameters, which is notably more than their baselines with 11 millions parameters. In use cases where the model needs to be deployed on edge devices with limited computational capacity or when the model is required to provide the results swiftly, the large amount of parameters and overall more computational power required can become a potential bottleneck.

1 https://github.com/BSI-OFIQ/OFIQ-Project - Accessed on 2024-03-29

As the objective for NeutrEx is to assess expression neutrality as a quality measurement, the proposed model is intended to fulfill all the requirements listed in the draft. However, a biometric system which fully conforms to the ISO/IEC 29794-5 would need to incorporate multiple quality components, many of which have not been developed yet or currently under development. From a practical perspective, deploying such system can be difficult, as the computational overhead resulted from combining different components with different architectures might increase significantly. Therefore, it is desirable to reduce the size of the singular component in multiple dimensions such as input size, processing time and storage capacity.

Our primary goal in this work is to optimize the NeutrEx solution to reduce the computational overhead by leveraging known methods in the field of neural network research. Initial literature research points to quantization [Guo18], pruning [Bla+20] and knowledge distillation [Gou+21] as possible methods that we investigate further in this work. As experimented by [HMD15], it is also possible to combine multiple techniques, for example pruning and quantization or pruning and knowledge distillation on the same architecture. In addition to implementing and experimenting on the methods above, it is also important that we can adequately benchmark the results by comparing them to the original NeutrEx.

1.2 GOALS

Motivated by our initial findings, we decide to implement a more lightweighted version of NeutrEx with focus on efficiency. The smaller version of NeutrEx - or **NeutrEx-lite** for brevity - is aimed to incorporate the optimization methods mentioned above to decisively reduce the computational overhead, but still maintain a competitive performance to its original counterpart. Formally, NeutrEx-lite intends to address the following research questions.

1. Which of the mentioned research directions: *Pruning*, *Quantization* and *Knowledge distillation* is most suitable to optimize NeutrEx while still maintaining the predictive capability of the quality measure?

2. How much can we improve NeutrEx's efficiency with the proposed solutions? What is the acceptable trade off in performance relative to reduced overhead in terms of number of parameters, storage space and inference time?

3. Would it be possible to combine multiple solutions?

2 LITERATURE RESEARCH

In this chapter, we introduce the key concepts to the NeutrEx framework by Grimmer et al. [Gri+23], which our work NeutrEx-lite is an extension of. In addition to that, we outline potential research directions and give a detailed overview on current literature for each method.

2.1 BACKGROUND KNOWLEDGE

2.1.1 *Monocular 3D Face Reconstruction*

Three-dimensional face modeling refers to the process of projecting a two-dimensional representation of a human face to a three-dimensional one, while trying to preserve as much details as possible. As such details are very important as a way to identify other human beings and process related information such as moods, expressions and intentions; there has always been a demand for better techniques in this field. Realistic and efficient reconstructions can be used in a wide range of applications; for example in the entertainment industry, in facial analysis research or in IT-Security. As pointed out in [Zol+18], 3D facial reconstruction can be categorized based on the quality of the capturing method, which often falls on one of the two approaches: Either the images are captured using the best equipment available and with more labor involved to ensure high quality; or they are provided by commercially available cameras that value performance and speed with a trade off in quality.

The state-of-the-art 3D face modeling method FLAME (**F**aces **L**earned with an **A**rticulated **M**odel and **E**xpressions) by [Li+17a] aims to address this gap in quality, with focus on capturing the variability in natural facial shape and expression. In FLAME, a 3D face is represented by a mesh consisting of 5,023 vertices; with 4 joints at neck, jaw and eyeballs which allow flexible adaptation of facial expressions. FLAME uses a combination of parameters to derive the mesh, namely the shape parameter β, pose parameter θ and expression parameter ψ. The authors train FLAME over 33,000 scans of 3D scans, and accomplish the goal of better, more realistic and more expressive face representation compared to other methods. There have been multiple researches that leverage the richness and distinctiveness of FLAME by inverting 2-dimensional face images into the FLAME parameter space, such as [DBB22; Fen+21; Pav+19; Zhe+22; Gra+22]. The NeutrEx framework makes use of two such researches, namely DECA [Fen+21] and EMOCA [DBB22].

DECA (**D**etailed **E**xpression **C**apture and **A**nimation) enables reconstruction and animation of face images in 3D. DECA introduces the novel *detail consistency loss*, which is calculated from multiple images of the same sub-

ject. This helps to disentangle static person-specific details from expression-dependent details during training, but requires additional face images from the same subjects for training. The coarse shape is first created by augmenting camera parameter c, albedo parameter α and light parameter l and combines them with the parameters β, θ and ψ from FLAME. All six parameters can be derived by the trained DECA encoder, which takes a 2D face image as input. The triple of parameters from FLAME is used to determine the coarse shape, while the new parameters are responsible for mapping the texture (e.g skin, appearance) of the subject into the coarse shape. A set of detail encoder and decoder is used to obtain the displacement map, which is then projected over the coarse shape to recreate the face image in 3D. For NeutrEx, the coarse shape reconstruction, i.e the reconstruction of the face in FLAME's model space, is needed. Therefore, Grimmer et al. incorporate the DECA encoder in their framework under the name *coarse shape encoder*.

EMOCA (**E**motion **D**riven **M**onocular Face **C**apture and **A**nimation) is an extension of DECA that focuses on capturing expression and emotional content by introducing *emotion consistency loss* which computes the differences in emotions between the input 2D image and rendered image by the model. EMOCA architecture consists of both DECA encoders and a new expression encoder, which is responsible for deriving expression parameter ψ separately from the DECA architecture. In this setup, it is possible to train the expression encoder independently from the DECA encoder, which loosens the requirements for training data and allows the model to learn from datasets with richer, more varied emotional expressions. This results in facial reconstruction that can convey more detailed expressions. The expression encoder of EMOCA is added to NeutrEx under the name *expression encoder* to derive the expression code ψ more correctly. In combination with the coarse shape encoder, this allows NeutrEx to accurately project 2D facial images into 3D face meshes.

2.1.2 *NeutrEx: A 3D Quality Component Measure on Facial Expression Neutrality*

As highlighted in previous chapter, NeutrEx is developed in compliance with the current draft international standard of ISO/IEC 29794-5 [Iso] with its main focus being able to assess expression neutrality. NeutrEx calculates how much a facial image deviates from the neutral expression anchor by utilizing 3D face technology. More concretely, for a given facial image, two monocular 3D face reconstruction encoders are used to derive necessary parameters for the reconstruction. The first encoder from DECA [Fen+21], maps the image to a coarse shape estimation. The second encoder from EMOCA [DBB22], extracts emotion-sensitive information from the image. The output parameters are then translated into the FLAME [Li+17b] parameter space for 2D-to-3D translation. To avoid the influence of identity-specific or pose-specific features, the face model is transformed to a generic head

shape and facing the camera frontally, i.e the corresponding parameters are set to zero. An overview of the framework is given in 2.1.

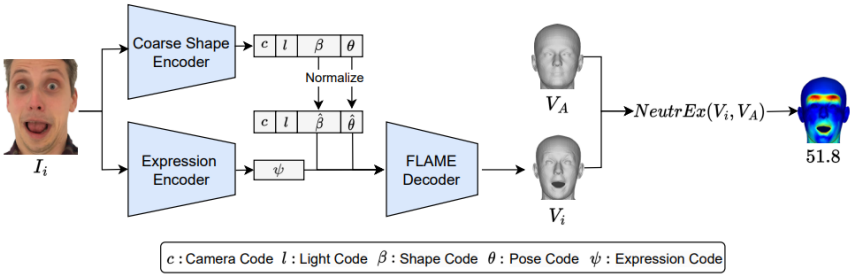

Figure 2.1: The NeutrEx [Gri+23] Pipeline: Facial images are encoded into the FLAME [Li+17b] parameter space using the coarse shape encoder from DECA [Fen+21] and the expression encoder of EMOCA [DBB22]. Source: [Gri+23].

To calculate the NeutrEx score, a neutral expression anchor is needed. The anchor for this approach is calculated by using the average expression code ψ_A from all images with neutral expression in a given dataset. Combined this expression code with the default parameters for head shape (β) and head pose (θ), the neutral expression anchor is reconstructed in FLAME space with 5,023 vertices. The algorithm iterates over all vertices from the reconstructed 3D facial image and calculate the per-vertex L2 distance

$$D(V_i) = ||V_i - V_A||_2 \tag{2.1}$$

between the reconstruction and the anchor, where V_i is the projection of the 2D facial image in 3D and V_A the neutral expression anchor calculated from the traing dataset. The NeutrEx score is then derived with min-max scaling from the maximum (D_{max}) and minimum (D_{min}) per-vertex distance measured based on the training dataset:

$$NeutrEx(V_i) = 100 * (1 - \frac{D(V_i) - D_{min}}{D_{max} - D_{min}}) \tag{2.2}$$

According to the authors, the effectiveness of NeutrEx measure as an utility predictor is evaluated by using the *Error-vs-Discard Characteristic (EDC)* curves following the guidelines specified by ISO/IEC 29794-1 [ISO23a]. By systematically removing images with the lowest NeutrEx score, the remaining image dataset should be able to achieve higher biometric prediction performance, i.e reduce the *false non-match rate* (FNMR) in a given dataset. Visually, the EDC curves should show a steep decline in steepness. Quantitatively, the algorithm with the best performance results in the lowest Partial Area Under Curve (pAUC). NeutrEx shows good results compared to other facial recognition utility predictors after benchmarking with two datasets Multi-PIE [Gro+10] and FEAFA+ [Gan+22], as shown in Figure 2.2 and 2.3.

With NeutrEx-lite, our main focus is to improve the efficiency of NeutrEx by reducing the number of parameters from the DECA and EMOCA

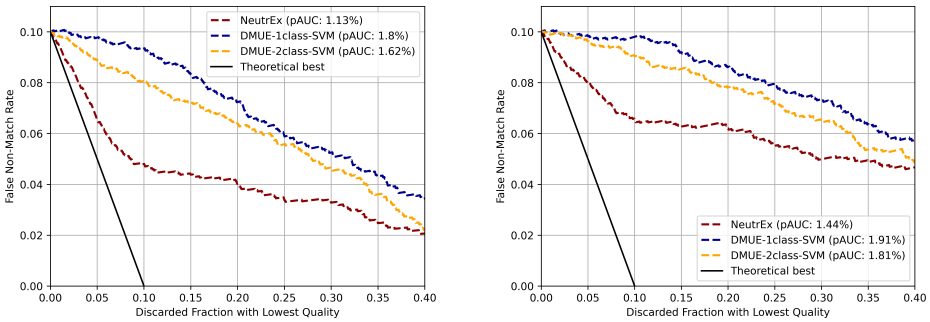

Figure 2.2: Performance evaluation of NeutrEx measure and baseline measures for predicting FR utility with MagFace [Men+21] (left) and Cognitec FaceVACS Version 9.6.0 (right) based on EDC curves using Multi-PIE dataset [Gro+10]. Source: [Gri+23].

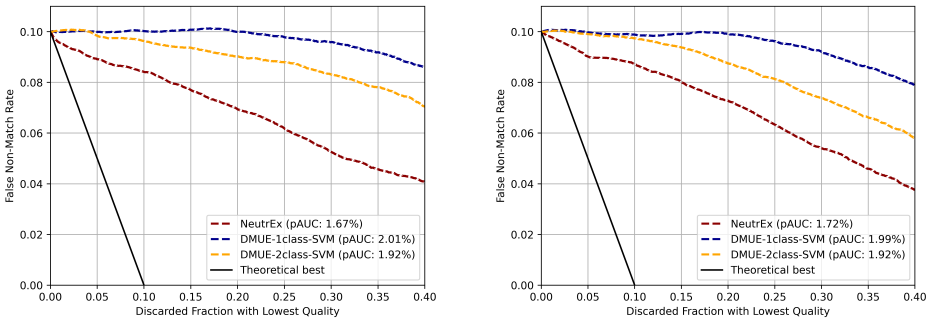

Figure 2.3: Performance evaluation of NeutrEx measure and baseline measures for predicting FR utility with MagFace [Men+21] (left) and Cognitec FaceVACS Version 9.6.0 (right) based on EDC curves using FEAFA+ dataset [Gan+22]. Source: [Gri+23].

encoders. We expect some performance trade-off by doing this, i.e the expected goal is not to improve the results from NeutrEx; but rather to remove as much parameters as possible, while still maintain a competitive performance to the original NeutrEx.

2.2 RESEARCH DIRECTIONS

Both encoders in NeutrEx are based on the ResNet-50 [He+15] architecture, which in turn is a form of deep convolutional neural network. We investigate the current methods in neural network research that can assist us with the main task of reducing the parameters in the encoders. The three methods of *pruning*, *quantization* and *knowledge distillation* have been extensively researched in recent years, which makes them possible candidates for our implementation. We give a detailed overview on each method and relevant publications in the subsequent sections.

2.2.1 *Pruning*

In neural network research, the term **pruning** refers to the process of systematically removing parameters from the network to reduce computational overhead and memory size [Bla+20]. After pruning, the desired outcome is a much more light-weight model, but still retains similar performance to the original one. This is based on the remark that neural networks are often over-parameterized [Han+15]. For example, [Den+13] observes that for a network with image-related task, the first layer usually consists of individual representation of each pixel. As it is likely that the value of one pixel correlate to its neighbor, the authors argue that this representation is a symptom of redundancy. In an extensive survey on pruning techniques, [Bla+20] concludes that pruning often leads to consistent decrease in computation needed and memory size. Motivated by these findings, our work investigates the effectiveness of pruning for NeutrEx model.

Pruning methods can be broadly categorized into two main approaches: **Unstructured pruning** and **structured pruning**. Unstructured pruning removes weights of the network individually across the whole network. Structured pruning on the other hand removes them on a per-group basis, i.e interconnecting elements of the network are removed altogether, which leads to changes in shapes and sizes of the model. Unstructured pruning leads to high compression rate, but requires specific support for realistic improvement, while unstructured pruning can be done with less support in terms of hardware and library [HX24].

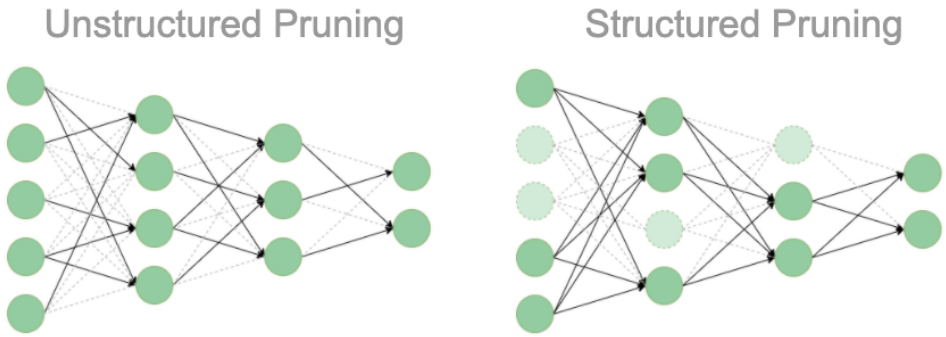

Figure 2.4: Visualization of unstructured pruning and structured pruning. Dashed elements represent removed network components. Source: [Mag].

Unstructured pruning ranks the individual weights of the network on some criteria, and then removing the redundant weights accordingly by setting them to zero. This leads to less hardware space required to store the model, and also reduces the computational time. [LDS89] and [HSW93] use the second-order Taylor expansion to approximate the parameter importance. [Lia+23; Min+18] use entropy to prune the model on a per-layer basis. [Han+15] only removes weights and neurons based on simple magnitude ranking, but introduces fine-tuning after removal to increase performance.

Figure 2.5: Example of a simple quantization method, where floating point 32 bits numbers are converted to 8 bits integer. Source: [DEVa].

Structured pruning removes groups of parameters, for example in channels [HZS17; Lin+20] or in convolutional layers [CZ18; CZM18]. Structured pruning can leverage methods from its unstructured counterpart and adapt them to meet the structural requirements [HX24]. [Li+16] removes filters and their connecting filters in subsequent layers based on the l_1-norm. Later implementations of this method also include l_2-norm as a criteria. [He+19] also prunes filters, but ranks them based on the geometric median. This is based on the assumption that on each layers; the filter which is closest to other filters geometrically contains information that can represented by those other filters instead, and thus can be removed without severely impacting the overall performance. [He+18] uses reinforcement learning to automatically determine pruning parameters in a structured setup without human intervention.

2.2.2 *Quantization*

Quantization aims to create a more compact version of a neural network by changing the representation of numerical values inside the model to more compacted format [Guo18]. Figure 2.5 shows a simplified example of quantization. A quantizied network is expected to have less storage requirement and more efficient processing speed, as simpler number format also allows less expensive computations. While a decrease in performance is expected, quantization is a potential research direction most suitable for stringent hardware requirements, such as when the targeted environment is on handheld devices where limited computational power and storage space are potential bottlenecks.

The most straightforward method of quantization is **rounding**, where the floating point numbers are replaced through rounding functions. Publications researching this method usually focus on deriving these function, and

redefine mathematical operations to make them more suitable for the new format. [Wu+18] proposes a framework which constraints weights, activations, gradients and errors among all layers to low bit-width integers in both training and inference. [CBD16] introduces a rounding function which reduces the number to a binary format and more importantly redefines all steps during training to accommodate the binary representation. [Das+18] proposes Dynamic Fixed Point (DFP) as a new format for floating point numbers in neural network, as well as all related mathematical operations. In a neural network using this format, all floating point numbers in a tensor share the same exponent. The exponent for each tensor is saved separately, and each floating point number in a tensor is represented as the mantissa only. In comparison with the IEEE-754 floating point numbers with the same bit length, DFP floats have higher precision due to the larger mantissa. According to the authors, the DFP-specific mathematical operations can potentially speed up the inference time by a factor of 2x.

Another quantization method is **vector quantization**. The weights are clustered and and replaced by the value of the cluster's centroid during reference. Each cluster is assigned an integer index, and the value of the corresponding centroid is stored somewhere else as a lookup table. By doing this, the precision is reduced from floating point to integer. Notable clustering methods include k-means [HMD15; Gon+14] and Hessian k-means clustering [CEKL16]. As noted by [Guo18], the computation of k-mean clustering is expensive, and using rounding methods when training from scratch is preferable to vector quantization.

Quantization can also be applied on a fully trained model instead of during training. This is called **post-training quantization**. Under the assumption that retraining or fine-tuning the models after quantization are not possible, [BNS19] introduces three methods to quantize the parameters in a trained model to a bit-width of 4 bits, while still maintains good accuracy compared to benchmarks. To address the significant inaccuracy observed in some methods after quantization, [Cho+19] builds a framework to find optimal quantization parameters at each layer of the network using minimum mean square error as the criteria. Notably, post-training quantization methods also gather industry interests [DEVa; DEVb; Doc], which can be seen as a result from increasing popularity of neural networks in mainstream usage.

2.2.3 *Knowledge Distillation*

Also known as teacher-student framework in some publications, **knowledge distillation** can be understood as a form of transfer learning, in which learned parameters from a source model (teacher model) can be leveraged during the training process of a different model (student model). After training, the student model is able to mimic the behaviors of the teacher given the same input, although a decrease in evaluation metrics is also expected. While transfer learning does not necessarily require that the resulting model has better performance w.r.t. storage or computational efficiency, these re-

quirements are the main goals for applying knowledge distillation in large neural network. Figure 2.6 from [Gou+21] shows a basic visualization of knowledge distillation in practice.

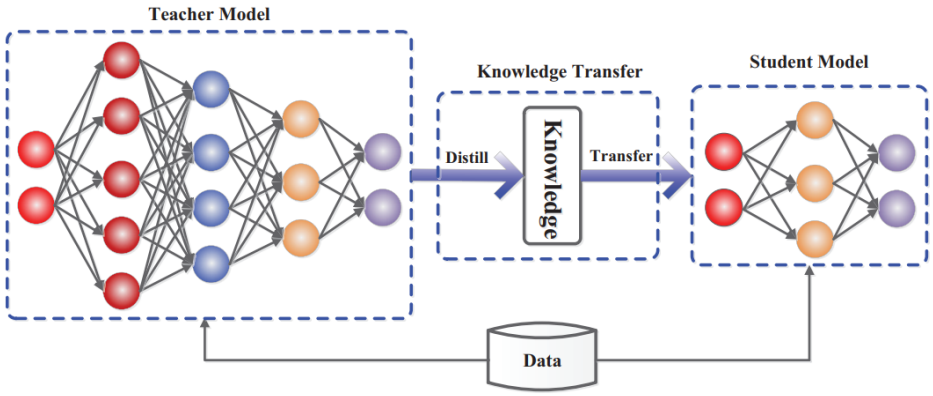

Figure 2.6: Teacher-student framework, also known as knowledge distillation. Source: [Gou+21]

The ideal outcome after training is that the student model can replicate the predictions of the teacher model given the same input data. Knowledge distillation has been applied with great success to large-scale neural network models in different disciplines, such as [Jia+19; Tan+19; San+19] in natural language processing or [Qia+18; Liu+18] for image recognition tasks. The reason for the surging popularity of knowledge distillation in recent years is the larger number of parameters and computational complexity of modern neural networks in many disciplines, and with it the trade-off in storage space and inference time.

Knowledge distillation can be applied during training of the teacher model, i.e the information is updated simultaneously for both models [Gou+21]. [Bou+22] distills the knowledge from the teacher to the student at multiple points while training the former. This allows the student to acquire knowledge continuously during the entire training phase, and enables it to learn more complex patterns from the teacher after both models converge. [Zha+18] proposes Deep Mutual Learning, in which multiple students learn from a pretrained teacher. During training, each student also acts as a teacher to other students, which ensures a wider variation of knowledge is passed to all students. [Wan+18] combines knowledge distillation with generative adversarial network, which simultaneously updates the teacher and the student with the discriminator's output.

In contrast to that, knowledge distillation can also be applied after the teacher is fully trained. In some cases, training a large neural network from scratch is infeasible given the unavailability of hardware, datasets or time constraints; which makes this distillation approach particularly attractive. [Jia+19] proposes a new method for distilling transformer-based network, which allows them to reduce a large scale pretrained network with 340 million parameters to 28% of its original size. [Mir+19] introduces an intermediate assistant teacher to bridge the gap in size between large pretrained

3 PREPARATIONS AND METHODOLOGY

In this chapter, we formally define our methodology and briefly describe our preparation for the experiments. We discuss the NeutrEx framework on multiple characteristics and define tangible targets for our NeutrEx-lite implementations. After that, we also introduce our benchmarking pipeline, which quantitatively compares NeutrEx-lite to NeutrEx on relevant metrics.

3.1 EVALUATING NEUTREX

We first take measurements of the two encoders in NeutrEx. We use the Python library `torchinfo`[1] to view the PyTorch[2] model's statistics, most importantly the overall precise number of parameters and total size on storage. In addition to that, we also report the estimated number of floating points operations (FLOPs) to process a randomly generated input. This number directly relate to the inference speed; since the more operations the model has to compute, the slower it can output a result. We believe that reporting this number is more convincing than a simple time measurement, since the actual inference time can be impacted by server capacity, etc. which are out of our control. We report the statistics in Table 3.1.

Name	**#. of parameters**	**FLOPs**	**Size on storage (MB)**
Coarse shape encoder (DECA)	25,848,108	4.09 billions	281.83
Expression encoder (EMOCA)	25,657,458	4.09 billions	281.06

Table 3.1: Number of parameters and size of two encoders according to `torchinfo`.

Figure 3.1 shows the baseline EDC curves from NeutrEx. As discussed earlier in Section 2.1.2, the primary method of benchmarking chosen by the authors of NeutrEx is the EDC curves on two datasets, Multi-PIE [Gro+10] and FEAFA+ [Gan+22]. From the benchmarking of NeutrEx, we observe that between predicting FR utility with MagFace [Gro+10] or Cognitec Face-VACS, there are no notable differences in terms of overall steepness as seen in Figure 2.2 and 2.3. Therefore, we decide to only use MagFace as the predictor for our experiment.

In addition to that, Figure 3.2 shows the class-wise distribution of NeutrEx score in correlation with the labels of images in the Multi-PIE and FEAFA+ datasets in kernel density estimation (KDE) plots. There are six expression

1 https://github.com/TylerYep/torchinfo
2 https://pytorch.org/

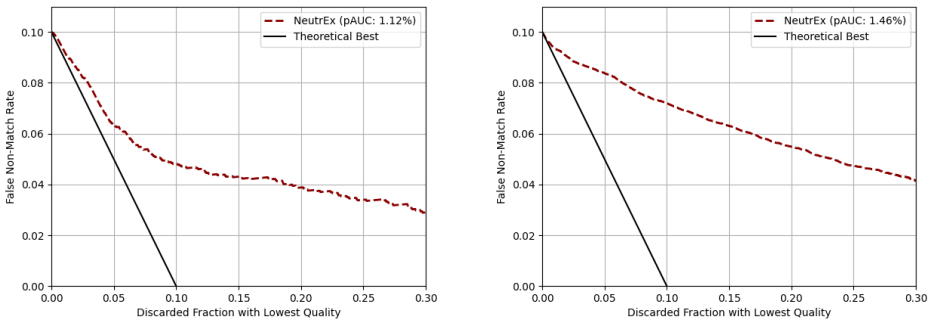

Figure 3.1: Baseline EDC curves of NeutrEx for predicting FR utility with MagFace on Multi-PIE [Gro+10] dataset (left) and FEAFA+ [Gan+22] (right).

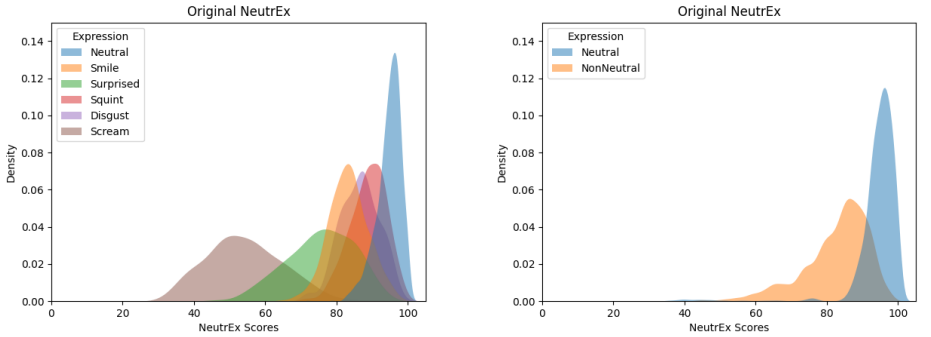

Figure 3.2: Class-wise distribution of NeutrEx scores in the datasets Multi-PIE (left) and FEAFA+ (right).

classes in Multi-PIE: *Neutral*, *Smile*, *Surprised*, *Squint*, *Disgust* and *Scream*. Looking at the distribution, we can see that the *Neutral* class is densely populated at higher scores. On one hand, expressions that are close to neutral such as *Smile* or *Squint* have higher scores and overlap with the distribution of *Neutral*. On the other hand, expressions that deviate significantly from neutral such as *Scream* or *Surprised* has lower NeutrEx scores and wider distribution, i.e they are more distant from a neutral state. The three labels *Smile*, *Squint*, *Disgust* also show significant overlap with each other. The same class-wise distribution graph can be plotted for the FEAFA+ dataset, although the images are only labeled binarily, i.e *Neutral* and *NonNeutral*. The same observation can be made about the *Neutral* class in this dataset, and the *Non-Neutral* one also trails behind in terms of scores like in Multi-PIE. We can also see that the overall NeutrEx score distributions for each dataset have some differences: In Multi-PIE, NeutrEx scores range from 25 to 100, while in FEAFA+ the scores range from 45 to 100. The overall distribution from both dataset gives an overview of how NeutrEx score functions as an expression neutrality predictor.

3.2 PREPARATIONS AND BENCHMARKING PIPELINE

In the context of NeutrEx-lite, we are interested in how it would compare with NeutrEx in the characteristics above. We experiment with the research directions mentioned in Section 2.2 to reduce the number of parameters in the encoders in NeutrEx. The results should ideally be smaller versions of the encoders, and we benchmark them against the performance of NeutrEx. First, the number of parameters as reported in Table 3.1 should be reduced, and with that also the storage size. Note that among the three research directions, pruning is the most straightforward method to accomplish this, and quantization can also be leveraged as a way to take advantage of the reduced number parameters to decrease inference time. Second, the smaller encoders should be competitive with NeutrEx when we compare them based on the EDC curves, i.e the newer curves should be close to NeutrEx's visually, and also maintain the steepness and pAUC to a degree. Third, we look at the distribution of expression classes in the evaluation datasets with the KDE plots. We are especially interested in the distribution of the *Neutral* classes in both datasets, as they are the most relevant to the purpose of NeutrEx and NeutrEx-lite; which is to measure facial expression neutrality. It would also be interesting to see if NeutrEx-lite can maintain the same characteristics as NeutrEx w.r.t to distribution of non-neutral classes, for example the overall distribution or the distribution within some classes. For the second and third comparisons, *knowledge distillation* stands out as a potential candidate, since the other two methods do not inherently involve training and re-calibration of models to increase task-specific performance. Figure 3.3 shows a comprehensive visualization of our benchmarking pipeline. Similar to NeutrEx, we also evaluate NeutrEx-lite on two datasets Multi-PIE and FEAFA+. The cross-database evaluation scheme gives us a good overview of how our solution behaves in different datasets.

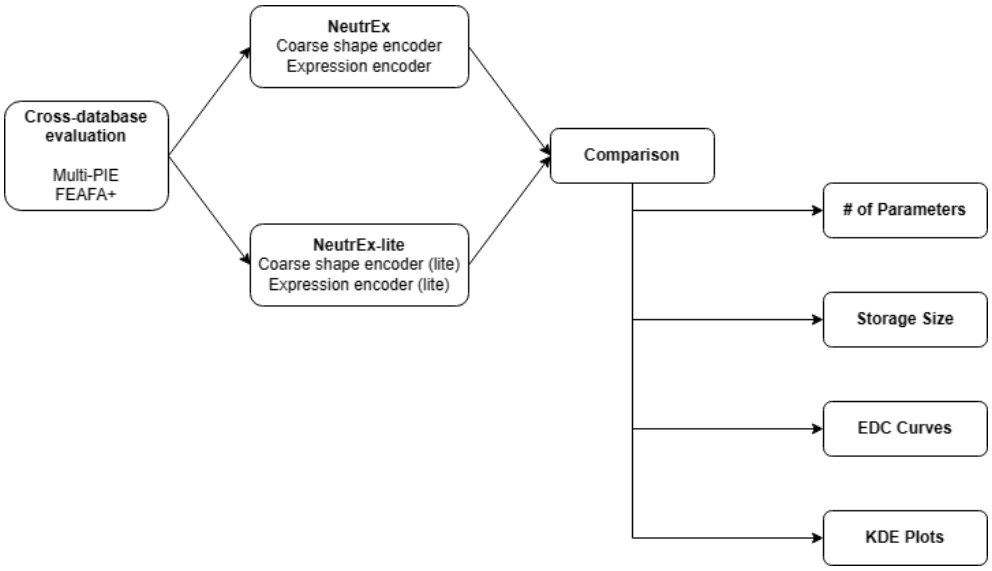

Figure 3.3: Benchmarking pipeline for NeutrEx-lite

We conduct all preparations, experiments and benchmarking on the GPU server of da/sec[3], which gives us access to a vGPU A100. In the original implementation of NeutrEx, the two encoders are accessed directly from the EMOCA [DBB22] model, more specifically the `EMOCA_v2_lr_mse_20` variant. This is possible because EMOCA is an extension of DECA which includes a fixed version of DECA's coarse shape encoder among other subnetworks. We disentangle the coarse shape encoder and the expression encoder from the model and save them locally. This allows us to make changes to the encoders without affecting other parts of the model, and also to investigate the effectiveness of each encoder in the NeutrEx pipeline independently: For each experiment, we only implement our changes on either the coarse shape encoder or the expression encoder, but not both of them at the same time. We use the smaller version of the encoder in combination with the untouched counterpart to calculate the NeutrEx scores for a dataset, and compare the results with NeutrEx's original results using the benchmark pipeline.

Since some experiments also require additional training and validation data, we look into datasets for expression neutrality after finalizing the benchmarking pipeline. We compose a dataset from various open-source facial expression datasets with different emotions. The majority of the images are taken in controlled environment, which points to a degree of uniformity w.r.t pose, lighting and distance between images of the same datasets and also between different datasets in the collection. The following datasets are included:

1. CelebA-HQ dataset. [Kar+17].
2. **C**hicago **F**ace **D**atabase (CFD) [MCW15].
3. Extended **C**ohn-**K**anade (CK+) dataset [Luc+10].
4. **F**lickr-**F**aces-**HQ** (FFHQ) dataset [KLA19].
5. **F**ace **R**ecognition **G**rand **C**hallenge (FRGCv2) dataset [Phi+05].
6. Multi-PIE dataset [Gro+10].
7. FEAFA+ [Gan+22].

We refer to this collection of datasets as **E**xpression **N**eutrality **D**ataset (END) in later reports.

In addition to that, we also use the AffectNet [MHM17] dataset for the fine-tuning experiments, since this is the same dataset that the authors of EMOCA used for training their model [DBB22]. Unlike END, the images in AffectNet are taken in-the-wild, and we notice there are some variations between images in terms of pose, lighting, distance to the camera and overall quality. AffectNet also has significantly more images than END, including up to ≈400k images compared to ≈7k images of END. Since AffectNet is not publicly available without the authors' permission, we do not include as part of our collection.

3 https://dasec.h-da.de/

4

PRUNING

We report the results of our pruning experiments in this chapter. We explain the used pruning strategies in Section 4.1. As explained in 3.2, we experiment on the encoders separately and report our findings in Section 4.2 and Section 4.3 respectively. The overall results are then discussed in 4.4.

4.1 INTRODUCTION

Pruning as a method to reduce parameters has been researched extensively in the academic setting [Lia+23; Min+18; HSW93; HZS17; Lin+20; Li+16; He+19] and in industrial setting [Int; Blo; PyT; Ten].

For our pruning experiments, we use the `Neural Network Intelligence`[1] to facilitate pruning. From the supported pruning algorithms (or pruners, as named by the developers of this library)[2], we choose three possible candidates: *L1 Norm pruner, L2 Norm pruner* and *FPGM (Filter Pruning via Geometric Median) pruner* and experiment with different settings. The reasoning behind the pruner selections is that they are implementations of influential pruning researches, which we mention in Section 2.2.1. All three implementations prune the model in a structured manner in which interconnecting filters are removed, which results in both storage space efficiency and inference speedup. The two pruners inspired by Li et al. [Li+16] prune filters from the convolutional layers by calculating the sum of kernel weights s_j, sort the filters accordingly and remove the ones with lowest ranking. We denote $F_{i,j}$ as the j_{th} filter of the i_{th} layer. L1 Norm pruner calculates s_j summing the absolute value:

$$s_j = \sum ||F_{i,j}||_1 \tag{4.1}$$

while L2 Norm pruner calculates the l_2-norm instead

$$s_j = \sum ||F_{i,j}||_2 \tag{4.2}$$

The geometric pruner implemented by He et al. [He+19] also works around filters in convolutional layers. First, the geometric median of a single layer is defined as the filter with the lowest total distance to every other filter in the layer:

1 https://github.com/microsoft/nni

2 https://nni.readthedocs.io/en/stable/compression/pruner.html - Accessed on 2024-03-30

$$x^{GM} = \arg\min_{x} \sum ||x - F_{i,j}||_2 \tag{4.3}$$

The filters nearest to the geometric median can be represented by other filters in the same layer, which makes pruning them have little impact to the overall performance. We prune all $F_{i,j*}$ that satisfies this constraint.

$$F_{i,j*} = \arg\min_{x} ||x^{GM} - F_{i,j}||_2 \tag{4.4}$$

To run the experiments, we first need to define some configurations. The most important parameter is the **sparse_ratio**, which is floating number between 0.0 and 1.0. The **sparse_ratio** defines the percentage of parameters to be removed, e.g a **sparse_ratio** of 0.1 removes 10% of parameters, and a **sparse_ratio** of 0.9 removes 90% of parameters. To maintain the consistency of the model in terms of shape and structure, we also group pruning targets in groups annotated with **dependency_group_id**. When a component is removed, every other components with the the same **dependency_group_id** are also removed as well. In practice, this results in additional removal when combined with the aforementioned **sparse_ratio**, i.e a ratio of 0.1 removes $10\% + x$ of parameters, with x being the amount of components linked to the removed 10%. Depending on structure of the model, this could lead to drastic removal of components.

Except for the output layer which needs to have the same shape as the original one, we allow all convolutional layers and linear layers of the ResNet-50 architecture to be pruned, i.e the encoder is pruned globally. Our experiments are influenced by two factors: First, which **pruner** out of L1 Norm pruner, L2 Norm pruner and FPGM pruner to use, and which **sparse_ratio** do we prune the encoder with. For brevity, we refer to our pruning experiments in later discussions in the following format:

`pruning-`*`pruner-encoder-sparse_ratio`*

where

- **pruner** can be `l1` for the L1 Norm pruner, `l2` for the L2 Norm pruner or `fpgm` for the FPGM pruner.
- **encoder** can be `cse` for the coarse shape encoder or `ee` for the expression encoder.
- **sparse_ratio** can be a floating number between 0.0 and 1.0. Initially, we prune between [0.3, 0.7] with an increment of 0.1

4.2 PRUNING THE COARSE SHAPE ENCODER

After pruning the coarse shape encoder, we run them through the proposed benchmarking pipeline in combination with a fixed expression encoder, calculate the NeutrEx scores for images in both Multi-PIE and FEAFA+ dataset;

and collect the reports for all of them. The statistics for pruning experiments with the L1 pruner, L2 pruner and FPGM pruner with sparse ratio between 0.3 and 0.7 (with an increment of 0.1 each) are tabulated in Table 4.1, 4.2 and 4.3 respectively. We report the number of parameters, size on storage, FLOPs and the pAUC of EDC curves in the tables.

Name	#. of parameters	FLOPs	Size (MB)	pAUC (%) Multi-PIE / FEAFA+
`l1-cse-0.3`	12,776,767	2.04 billion	176.59	1.26 / 1.32
`l1-cse-0.4`	9,415,422	1.51 billion	145.46	1.25 / 1.31
`l1-cse-0.5`	6,538,508	1.05 billion	115.67	1.25 / 1.30
`l1-cse-0.6`	4,230,167	692.41 million	89.14	1.24 / 1.31
`l1-cse-0.7`	2,404,745	401.50 million	64.15	1.25 / 1.31

Table 4.1: Pruning the coarse shape encoder with L1 Norm pruner. The `pruning-` prefix is omitted from the name in this Table for brevity.

Name	#. of parameters	FLOPs	Size (MB)	pAUC (%) Multi-PIE / FEAFA+
`l2-cse-0.3`	12,776,767	2.04 billion	176.59	1.22 / 1.36
`l2-cse-0.4`	4,665,594	756.12 million	94.26	1.25 / 1.31
`l2-cse-0.5`	1,200,776	204.25 million	43.17	1.25 / 1.31
`l2-cse-0.6`	213,273	41.01 million	16.96	1.25 / 1.31
`l2-cse-0.7`	26,956	6.89 million	5.73	1.19 / 1.28

Table 4.2: Pruning the coarse shape encoder with L2 Norm pruner. The `pruning-` prefix is omitted from the name in this Table for brevity.

Name	#. of parameters	FLOPs	Size (MB)	pAUC (%) Multi-PIE / FEAFA+
`fpgm-cse-0.3`	12,776,767	2.04 billion	176.59	1.25 / 1.31
`fpgm-cse-0.4`	4,665,594	756.12 million	94.26	1.25 / 1.31
`fpgm-cse-0.5`	1,200,776	204.25 million	43.17	1.25 / 1.31
`fpgm-cse-0.6`	213,273	41.01 million	16.96	1.23 / 1.30
`fpgm-cse-0.7`	26,956	6.89 million	5.73	1.19 / 1.28

Table 4.3: Pruning the coarse shape encoder with FPGM pruner. The `pruning-`prefix is omitted from the name in this Table for brevity.

All pruners succeed in reducing the number of parameters of the coarse shape encoder significantly. However, the experiments with L1 Norm pruner remove considerably fewer parameters compared to its peer at the same **sparse_ratio**, for example $\approx$ 6 million parameters remain at 0.5 ratio compared to $\approx$ 1.2 million parameters for both L2 and FPGM pruners. Noticeably, L2 and FPGM pruners report the same number of parameters remain for all experimented ratios, although comparisons in other characteristics

show that there are some differences between the resulting encoders. At 0.7 ratio, both pruners reduce the coarse shape encoder to 26 thousand parameters, which means 99,9% of parameters get removed. This is a drastic removal of components, which can be attributed to the chosen method of pruning and the respective pruners.

Thanks to the reduced number of parameters; we also observe that it takes a lot less time for inference, both in the actual time it takes for the experiment to benchmark in our experience and in the reported number of addition and multiplication operations. The storage size also gets smaller, down to 5MB in the best case which can easily be loaded into edge devices with lower storage capacity.

Next, we look at the EDC curves when using the pruned coarse shape encoder. Since there are a lot of results available, we only discuss a few standout experiments in this section. We discuss the `pruning-l1-cse-0.3` and `pruning-l1-cse-0.7`; as they are the versions with the least and the most pruned parameters respectively while do not have their parameters drastically removed. Figure 4.1 and 4.2 show the EDC curves for the two experiments.

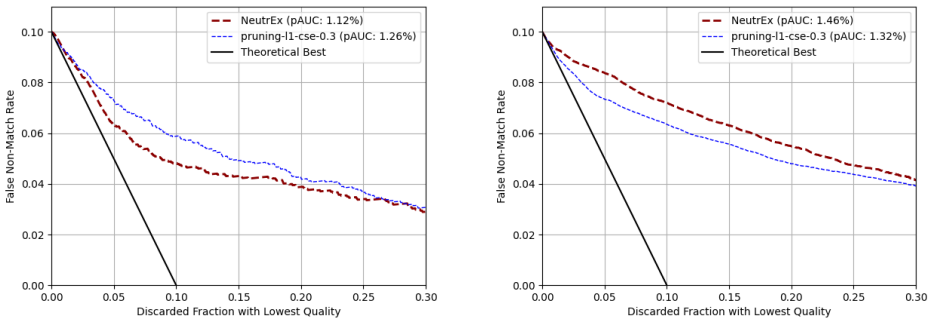

Figure 4.1: EDC curves for `pruning-l1-cse-0.3` benchmarked with Multi-PIE (left) and FEAFA+ (right)

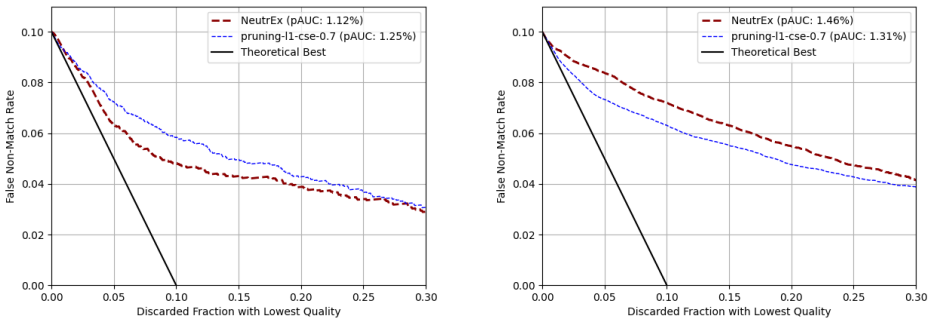

Figure 4.2: EDC curves for `pruning-l1-cse-0.7` benchmarked with Multi-PIE (left) and FEAFA+ (right)

In the 0.3 variant, the new coarse shape encoder shows a slight decrease in performance when benchmarked with Multi-PIE compared to original Neu-

trEx. In the range of between 0% and 30% of discard samples with expressions that deviate the most from neutrality, our experiment reports a higher pAUC (1.26%) compared to NeutrEx (1.12%), and we also have worse false non-match rate (FNMR) at every discarded fraction up to 30%. However, we still maintain the overall steepness of the EDC curve, and towards the higher end of the discard fraction our performance gets noticeably closer to NeutrEx. On the contrary, our coarse shape encoder outperforms NeutrEx when benchmarked FEAFA+ dataset. Under the same assumption, this experiment reports lower pAUC (1.32%) compared to NeutrEx (1.46%) and a lower FNMR at every discarded fraction. This is a surprising result, as we expect performance to trend downwards after removal of parameters, not a slight increase as in this case. We can make the same observations for the 0.7 variant, as visually the EDC curves are very similar, and the reported pAUC only have a difference of 0.01%.

Next, we compare the class-wise distributions of our pruned encoders with that of NeutrEx. Figure 4.3 shows the performance of the 0.3 variant with the Multi-PIE dataset. The overall score distribution gets significantly pushed to the right, i.e our experiment gives higher scores to all samples regardless of the actual label. As a consequence, there's a lot more overlapping between different expressions in terms of NeutrEx score. Looking at each label independently, this pruned encoder keeps the distribution characteristics for the majority of labels. For example, the *Scream* expression stills receive relatively lower scores; and the triple of labels *Smile, Squint* and *Disgust* still significantly overlap with each other. Most importantly, the distribution of the *Neutral* class stays mostly the same, which is the most relevant characteristic for NeutrEx and NeutrEx-lite. The only label with significant changes is the *Surprised* label, as it is given a higher score and also a higher density compared to the original. We also receives similarly good performance on the FEAFA+ dataset (Figure 4.4): Despite the overall higher score, the *Neutral* and *NonNeutral* remain relatively distinct, and the distribution of *Neutral* stays unchanged. The large shift of score distribution to the right indicate a lower performance in utility prediction tasks where we want to discard images with lower scores as they contribute less to the system.

The class-wise distribution of the 0.7 variant compared to original NeutrEx and the 0.3 variant is shown in Figure 4.3 for Multi-PIE dataset, and Figure 4.4 for FEAFA+. Surprisingly, both variants display almost identical characteristics like in our previous assessment with the EDC curves. Our expectation before starting the experiments is with the lower number of parameters, the performance would steadily decrease. Instead, the performance seems to be largely unchanged between two variants, regardless of how many parameters get removed. To further investigate this, we take a closer look at the performance of other coarse shape encoder when pruned with the same L1 Norm pruner, i.e the variants with 0.4, 0.5 and 0.6 sparse ratios which reveals similar observations in both the EDC curves 4.5 and KDE distributions 4.6, i.e there are no discernible differences when changing the sparse ratio.

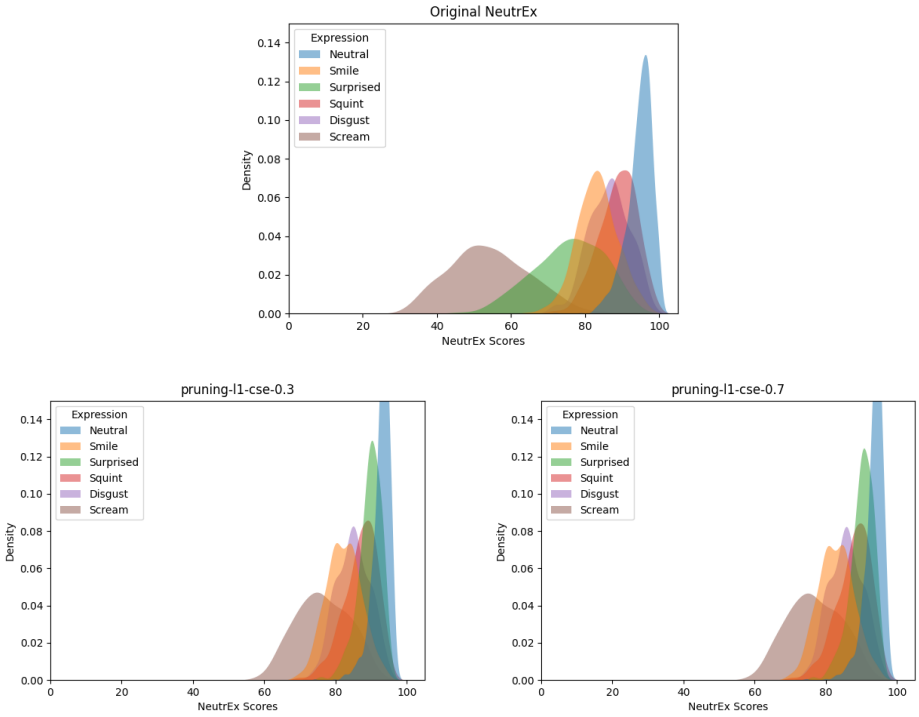

Figure 4.3: Class-wise distributions comparison between NeutrEx (top), `pruning-l1-cse-0.3` (bottom, left) and `pruning-l1-cse-0.7` (bottom, right). Benchmarked with Multi-PIE.

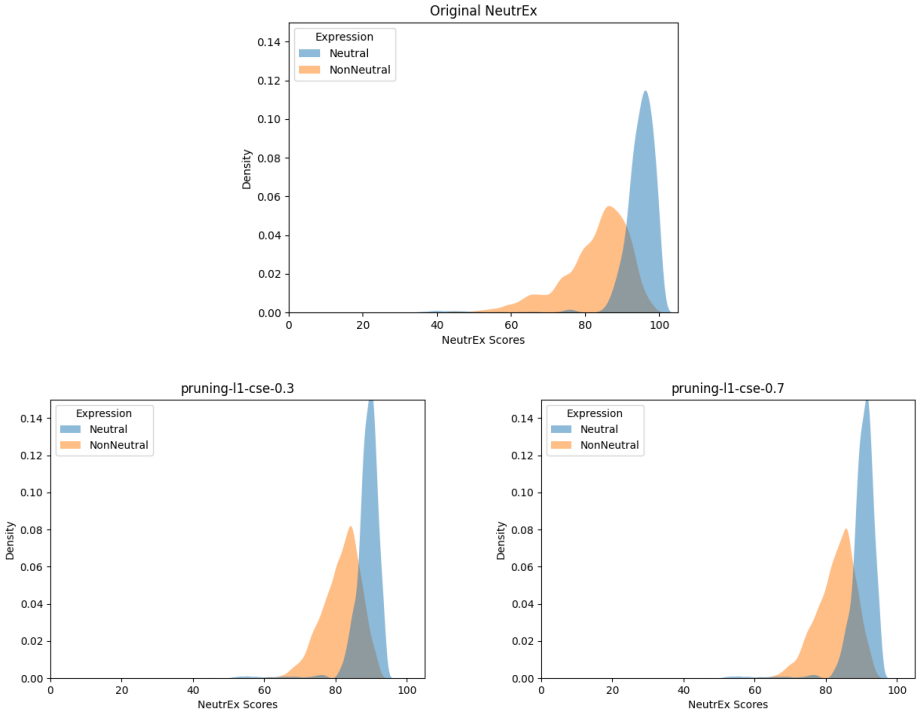

Figure 4.4: Class-wise distributions comparison between NeutrEx (top), `pruning-l1-cse-0.3` (bottom, left) and `pruning-l1-cse-0.7` (bottom, right). Benchmarked with FEAFA+.

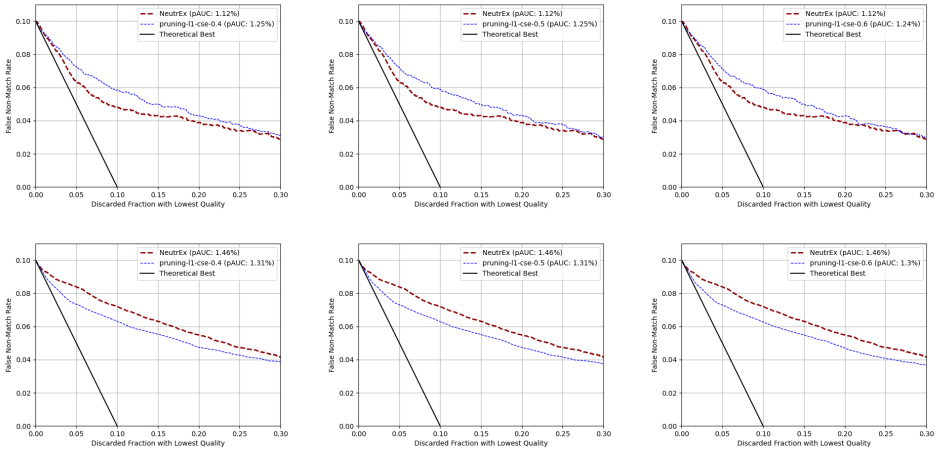

Figure 4.5: EDC curves for `pruning-l1-cse-0.4` (left), `pruning-l1-cse-0.5` (middle) and `pruning-l1-cse-0.6` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

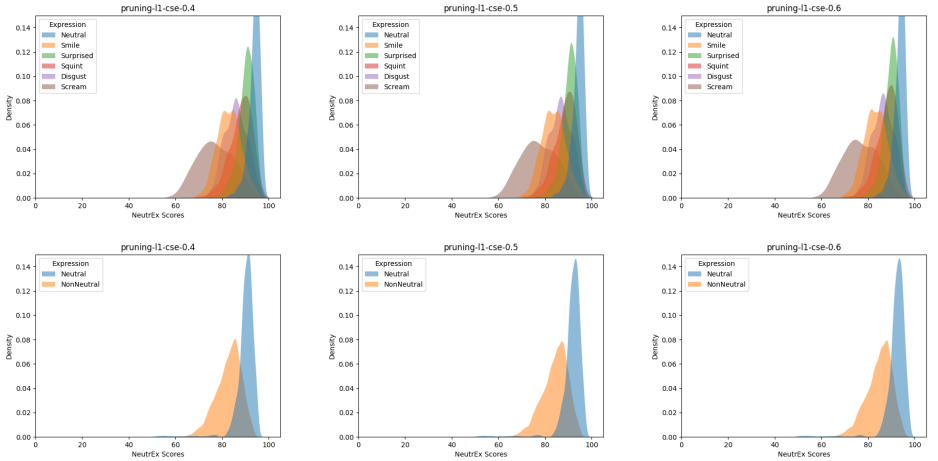

Figure 4.6: Class-wise distributions for `pruning-l1-cse-0.4` (left), `pruning-l1-cse-0.5` (middle) and `pruning-l1-cse-0.6` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

While the results themselves are encouraging and achieve our targets, this points to a potential problem as it does not correlate with our understanding of the pruning approach. In our assessment, there are two possible causes to this:

1. The L1 Norm pruner is ineffective at pruning the model accurately.
2. The defined sparse ratio is ineffective at reducing the number of parameters reasonably.

We investigate the first possible issue by looking at other experiments utilizing other pruners, namely the L2 Norm pruner and the FPGM pruner. For both pruners, we observe more changes in EDC curves and KDE distributions compared to L1 Norm pruner. However, going from lower sparse ratio to a higher one still shows no significant changes in performance similar to our previous experiments. As an example, we show comparisons between `pruning-l2-cse-0.3` and `pruning-l2-cse-0.7` in Figure 4.7 and Figure 4.8; as well as between `pruning-fpgm-cse-0.3` and `pruning-fpgm-cse-0.7` in Figure 4.9 and Figure 4.10. With this, we conclude that the issue is not exclusive to a pruner only, but affect all our chosen pruners to some degree.

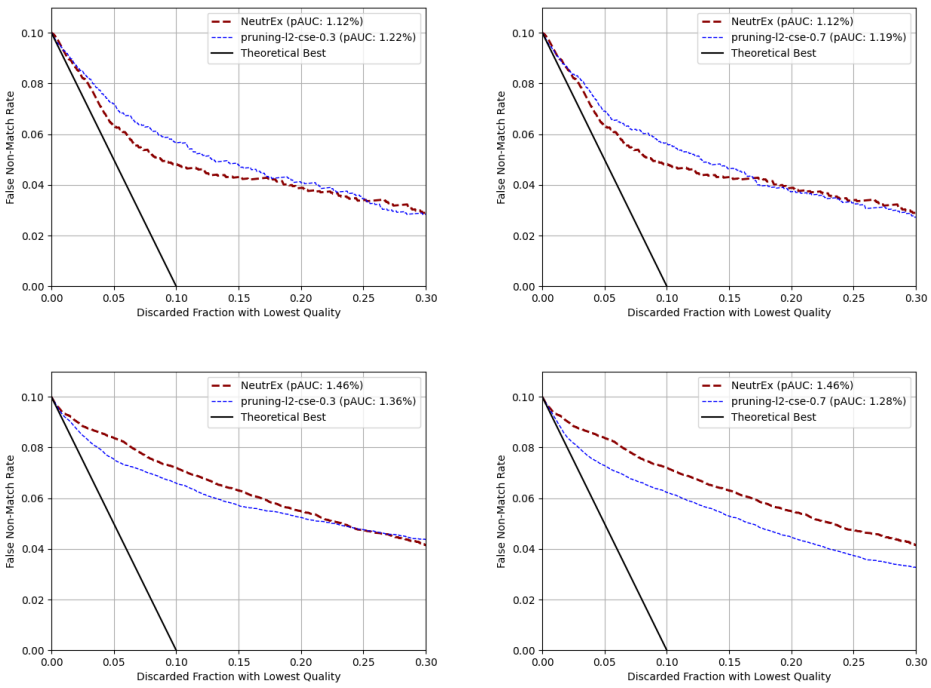

Figure 4.7: EDC curves for pruning experiment `pruning-l2-cse-0.3` (left) and `pruning-l2-cse-0.7` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

Next, we look into the second possible issue by conducting pruning experiments with lower sparse ratios: 0.1, 0.15 and 0.2. While the experiments with 0.15 and 0.2 show the same symptoms as previous ones with higher sparse

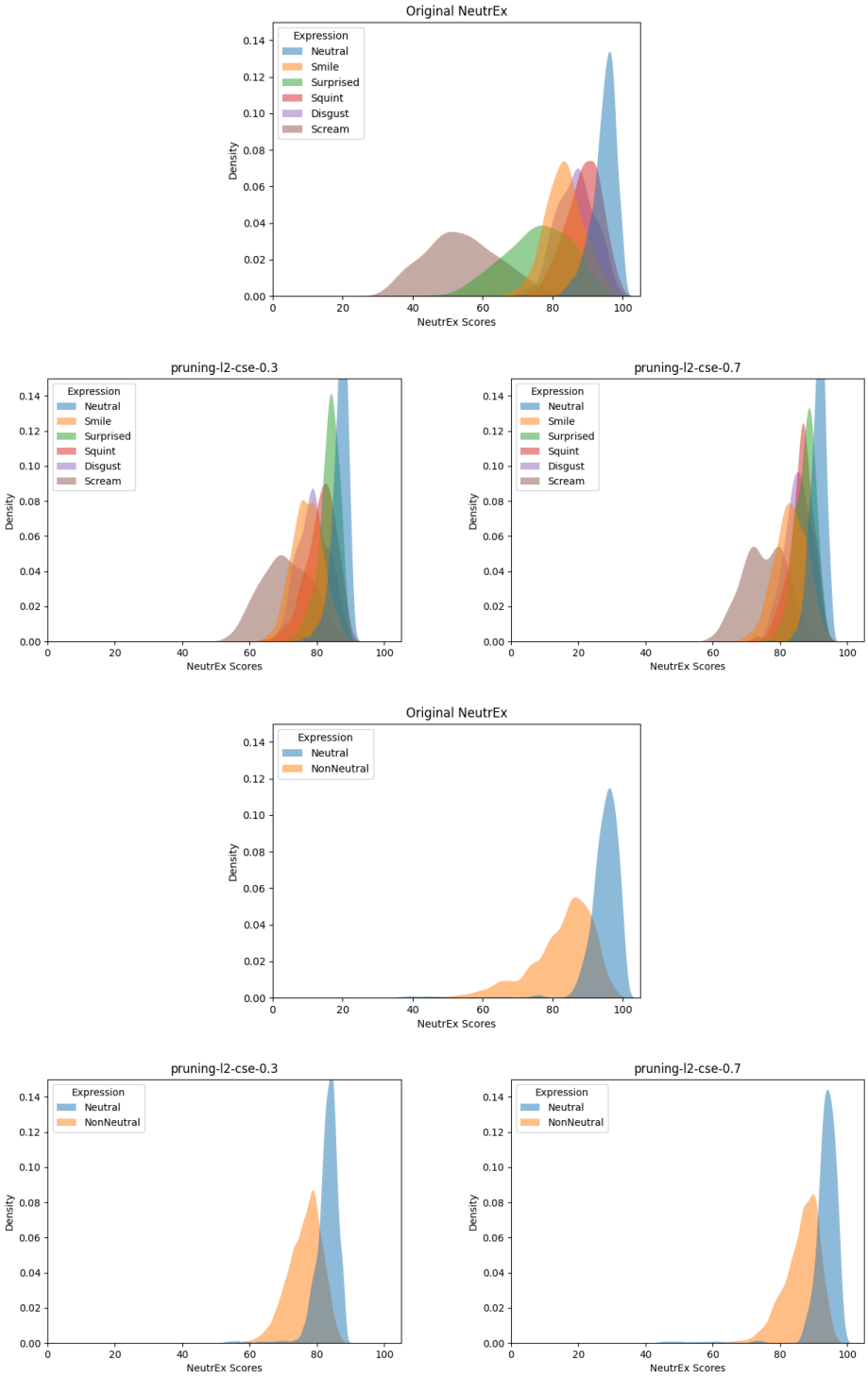

Figure 4.8: Class-wise distributions for `pruning-l2-cse-0.3` (left) and `pruning-l2-cse-0.7` (right). Benchmarked with Multi-PIE (top 3 rows) and FEAFA+ (bottom 3 rows).

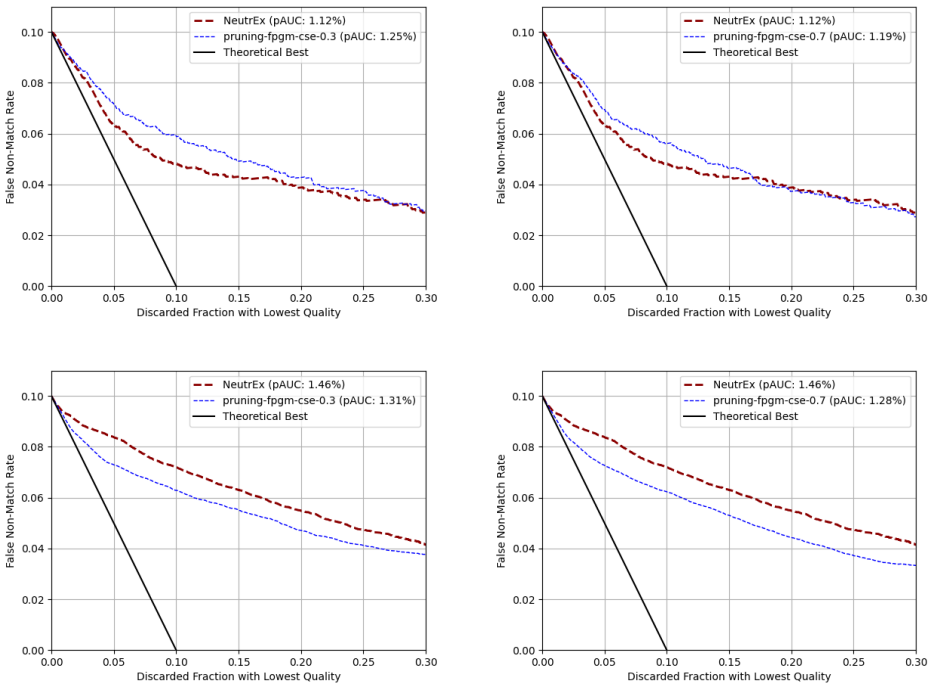

Figure 4.9: EDC curves for `pruning-fpgm-cse-0.3` (left) and `pruning-fpgm-cse-0.7` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

ratio, experiments with 0.1 sparse ratio perform very closely to NeutrEx as seen in Figure 4.11, 4.12 and 4.13 for all pruners.

More importantly, we can see a significant drop in performance when going from 0.1 sparse ratio to higher ratio, especially when we look at the KDE distributions in both datasets. This is in line with our understanding of pruning and its effect on performance of neural network model. We hypothesize that the similarity in performance is related to the ranking of parameters in the chosen pruning methods. All of our pruners rank the parameters based on some criterion to reflect their importance, and then remove parameters considered to be least important. Since the pruning is carried out deterministically, the most important parameters for inference are ranked higher and remain the same in all variants, even with the increasing sparse ratio. As they are not changed or updated in anyway, we receive similar performance reports for almost all of our experiments after pruning. This puts an emphasis on finding the suitable sparse ratios in our pruning experiments going forward, in order to avoid over-removing parameters from the models. To complete our experiments with 0.1 sparse ratio, we report the number of parameters and related characteristics in Table 4.4.

Our experiments show that pruning the coarse shape encoder is a viable method to reduce the number of parameters in NeutrEx. We summarize our findings as follow:

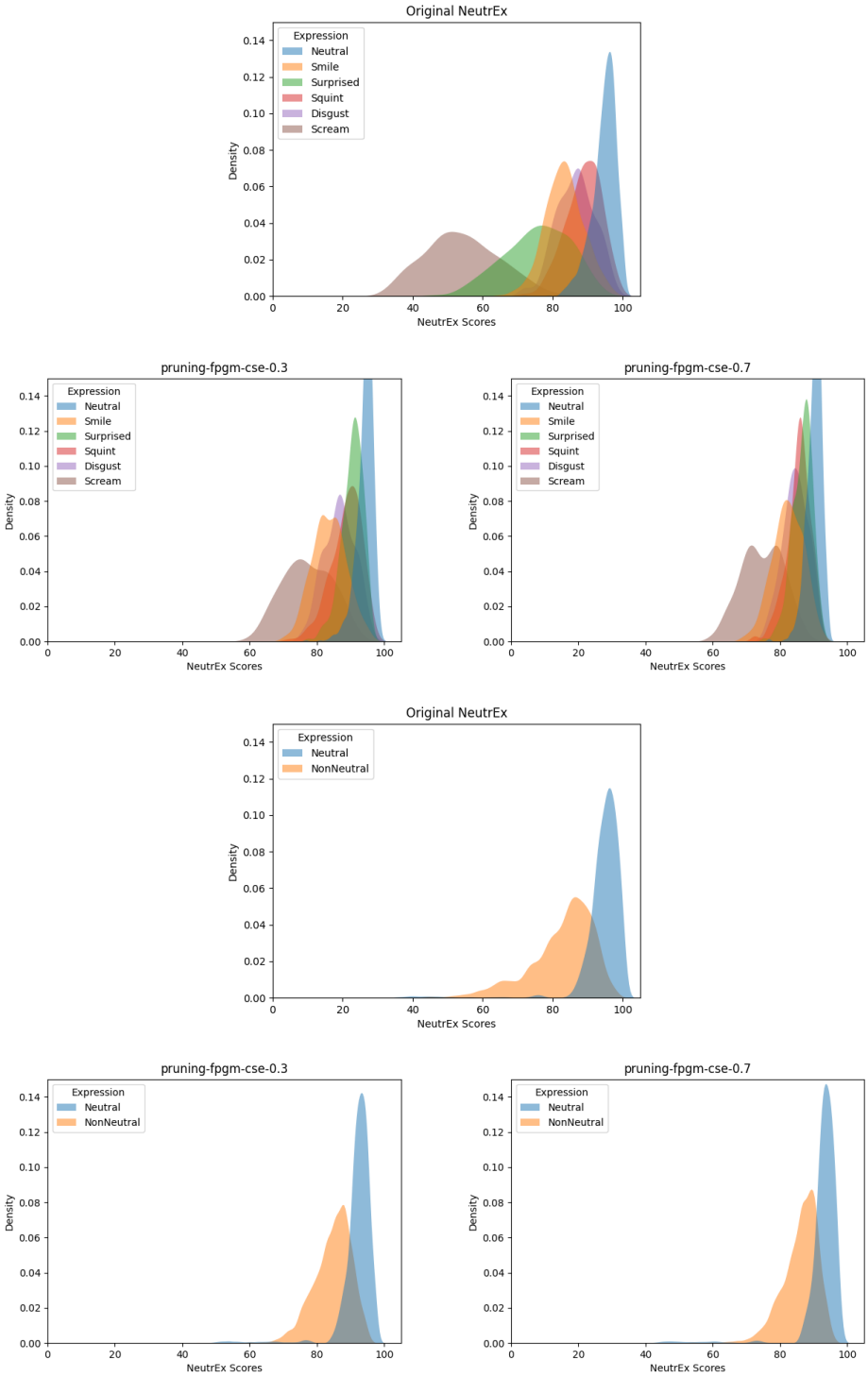

Figure 4.10: Class-wise distributions for `pruning-fpgm-cse-0.3` (left) and `pruning-fpgm-cse-0.7` (right). Benchmarked with Multi-PIE (top 3 rows) and FEAFA+ (bottom 3 rows).

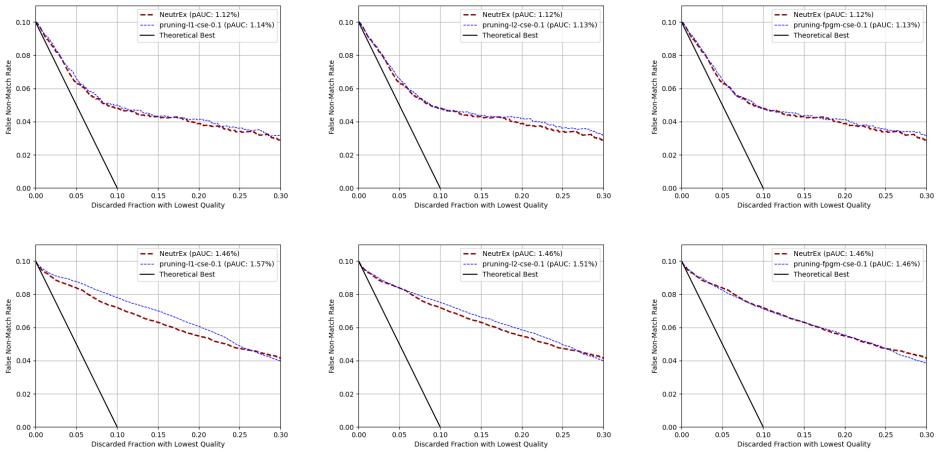

Figure 4.11: EDC curves for `pruning-l1-cse-0.1` (left), `pruning-l2-cse-0.1` (middle) and `pruning-fpgm-cse-0.1` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

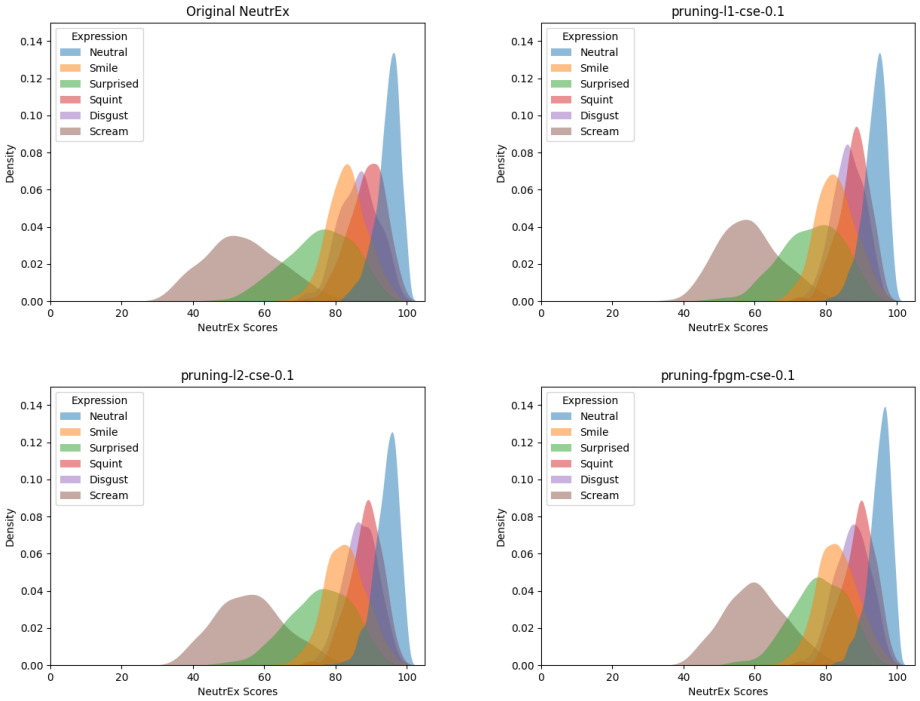

Figure 4.12: Class-wise distributions for `pruning-l1-cse-0.1` (top, right), `pruning-l2-cse-0.1` (bottom, left) and `pruning-fpgm-cse-0.1` (bottom,right). NeutrEx (top, left) for comparison. Benchmarked with Multi-PIE.

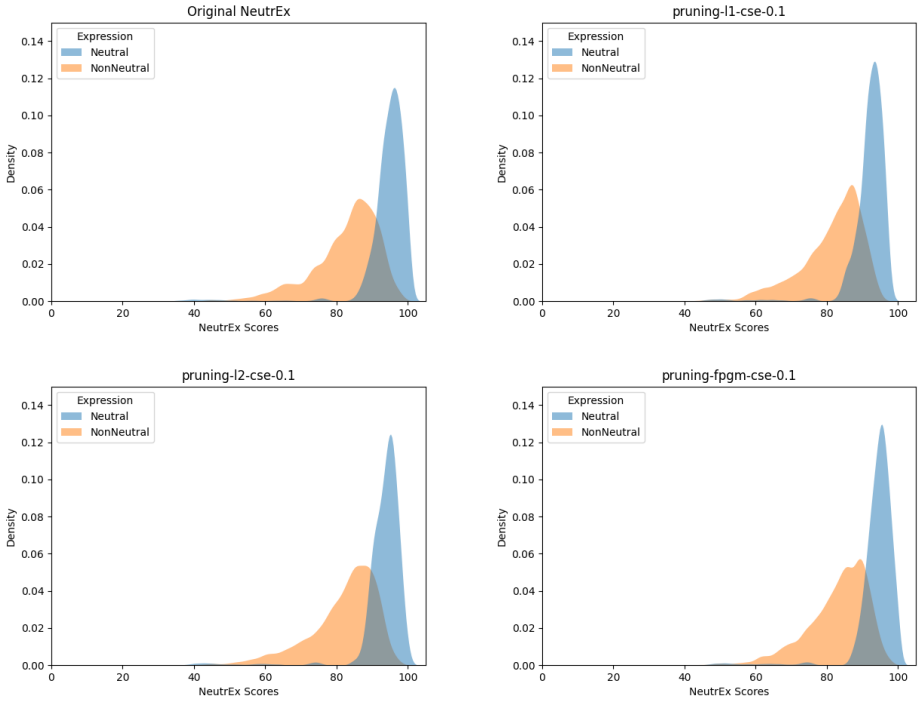

Figure 4.13: Class-wise distributions for `pruning-l1-cse-0.1` (top, right), `pruning-l1-cse-0.1` (bottom, left) and `pruning-l1-cse-0.1` (bottom,right). NeutrEx (top, left) for comparison. Benchmarked with FEAFA+.

Name	#. of parameters	FLOPs	Size (MB)	pAUC (%) Multi-PIE / FEAFA+
`l1-cse-0.1`	21,011,447	3.34 billion	245.18	1.14 / 1.57
`l2-cse-0.1`	21,011,447	3.34 billion	245.18	1.13 / 1.51
`fpgm-cse-0.1`	21,011,447	3.34 billion	245.18	1.13 / 1.46

Table 4.4: Pruning the coarse shape encoder at 0.1 sparse ratio with all pruners. The `pruning`-prefix is omitted from the name in this Table for brevity.

1. Among the three pruners, L1 Norm pruner removes fewer parameters compare to L2 Norm pruner and FPGM pruner. However, the choice of pruner does not have a significant impact on the overall results.

2. There is a decrease in performance when we increase the sparse ratio. At some point, the performance converges i.e removing more parameters no longer shows changes when benchmarked with our pipeline.

3. The experiments with 0.1 sparse ratio are closest to NeutrEx when we compare the EDC curves and class-wise distribution.

4. Even with the majority of parameters removed as shown in experiments `pruning-fpgm-cse-07` and `pruning-l2-cse-07`, we still manage to keep a reasonably close performance to NeutrEx.

4.3 PRUNING THE EXPRESSION ENCODER

After pruning the coarse shape encoder, we turn our attention to the expression encoder. Similar to previous experiments; we perform changes on the expression encoder, and combine the pruned encoder with the original coarse shape encoder extracted from EMOCA. We benchmark them with the same pipeline as proposed before. In addition to that, we make two changes to our experiment setup based on the information that we learn from the coarse shape encoder experiments:

1. Prune the encoders with ratio from 0.1 to 0.7 (with an increment of 0.1) instead of from 0.3 to 0.7.

2. Remove the L2 Norm pruner as a pruner option, since its performance is almost identical to the FPGM one.

In Table 4.5 and 4.6, we report the number of parameters remaining in the expression encoder after pruning, as well as the size on storage and estimated number of mathematical operations. Similar to previous experiments with coarse shape encoder, the number of parameters are reduced significantly. On average, slightly fewer parameters are retained for the pruned expression encoder than for the coarse shape encoder. At higher ratios, the encoder is pruned more aggressively which leads to more than 80% of the parameters removed. Both pruners report mostly the same number of parameters at each corresponding ratio. Experiment `pruning-l1-ee-0.2` and `pruning-l1-ee-0.3` standout in the report for their unusual results. Despite having a higher sparse ratio, `pruning-l1-ee-0.3` retains $\approx 5\%$ more parameters than `pruning-l1-ee-0.2`. We suspect that the issue is with `pruning-l1-ee-0.3`, and discuss this in details later in this Section.

As with the coarse shape encoder experiments, we look into the EDC curves and class-wise distributions to evaluate the performance of our experiments. Due to the high number of experiments, we only discuss the most insightful results in this Section. Figure 4.14, 4.15 and 4.16 show how the encoders pruned with 0.1 sparse ratio perform on our benchmarks.

Name	#. of parameters	FLOPs	Size (MB)	pAUC (%) Multi-PIE / FEAFA+
l1-ee-0.1	20,839,769	3.34 billion	244.50	1.15 / 1.53
l1-ee-0.2	11,953,230	1.94 billion	170.02	1.16 / 1.46
l1-ee-0.3	12,643,219	2.04 billion	176.06	1.27 / 1.78
l1-ee-0.4	4,585,242	756.04 million	93.94	1.17 / 1.75
l1-ee-0.5	1,160,414	204.21 million	43.00	1.17 / 1.76
l1-ee-0.6	196,905	40.99 million	40.99	1.17 / 1.74
l1-ee-0.7	21,748	6.89 million	5.71	1.17 / 1.74

Table 4.5: Pruning the expression encoder with L1 Norm pruner. The pruning-prefix is omitted from the name in this Table for brevity.

Name	#. of parameters	FLOPs	Size (MB)	pAUC (%) Multi-PIE / FEAFA+
fpgm-ee-0.1	20,839,769	3.34 billion	244.50	1.17 / 1.50
fpgm-ee-0.2	13,375,028	2.17 billion	182.93	1.17 / 1.86
fpgm-ee-0.3	12,643,219	2.04 billion	176.06	1.17 / 1.76
fpgm-ee-0.4	4,585,242	756.04 million	93.94	1.17 / 1.75
fpgm-ee-0.5	1,160,414	204.21 million	43.00	1.17 / 1.74
fpgm-ee-0.6	196,905	40.99 million	40.99	1.17 / 1.74
fpgm-ee-0.7	21,748	6.89 million	5.71	1.17 / 1.74

Table 4.6: Pruning the expression encoder with FPGM pruner. The pruning-prefix is omitted from the name in this Table for brevity.

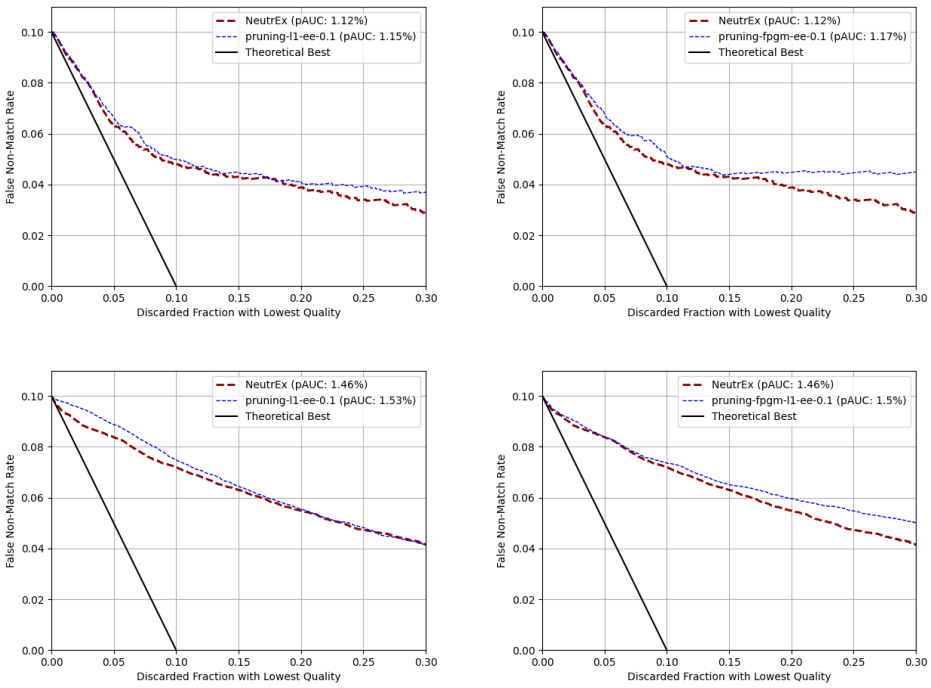

Figure 4.14: EDC curves for `pruning-l1-ee-0.1` (left) and `pruning-fpgm-ee-0.1` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

In the EDC curves (Figure 4.14), we can see that both of our experiments perform well when benchmarked with the FEAFA+ dataset; with `pruning-l1-ee-0.1` perform better at higher discarded fraction, and `pruning-fpgm-ee-0.1` perform better at lower discarded fraction. In average, they report a pAUC of 1.51 % compared to NeutrEx's 1.46 % which is encouraging. Our performance is also competitive in the Multi-PIE dataset, reporting 1.15% pAUC and 1.17% pAUC respectively against NeutrEx's 1.12%. However, our experiment's performance stagnate after the 0.15% fraction, as the false non-match rate does not go down with the increasing discarded fraction.

Next, we look into the class-wise distribution when evaluated with the Multi-PIE dataset (Figure 4.15). Both encoders maintain the overall distribution, but the class-wise performance is noticeably worse. In `pruning-l1-ee-0.1`, there is a large overlap between *Neutral*, *Squint* and *Disgust*. While *Neutral* still maintains its density and score distribution compared to NeutrEx, *Squint* and *Disgust* receive significantly higher score. The label *Smile* is also more densely concentrated, but the score distribution remains relatively the same. Expressions that are distant from *Neutral* such as *Surprised* and *Scream* show little changes in both density and distribution. `pruning-fpgm-ee-0.1` exacerbates the overlapping problem even more, as *Neutral*, *Squint* and *Disgust* all show the same distribution and density. The overlapping of *Neutral* with other classes poses a serious problem for NeutrEx in neutrality estimation tasks.

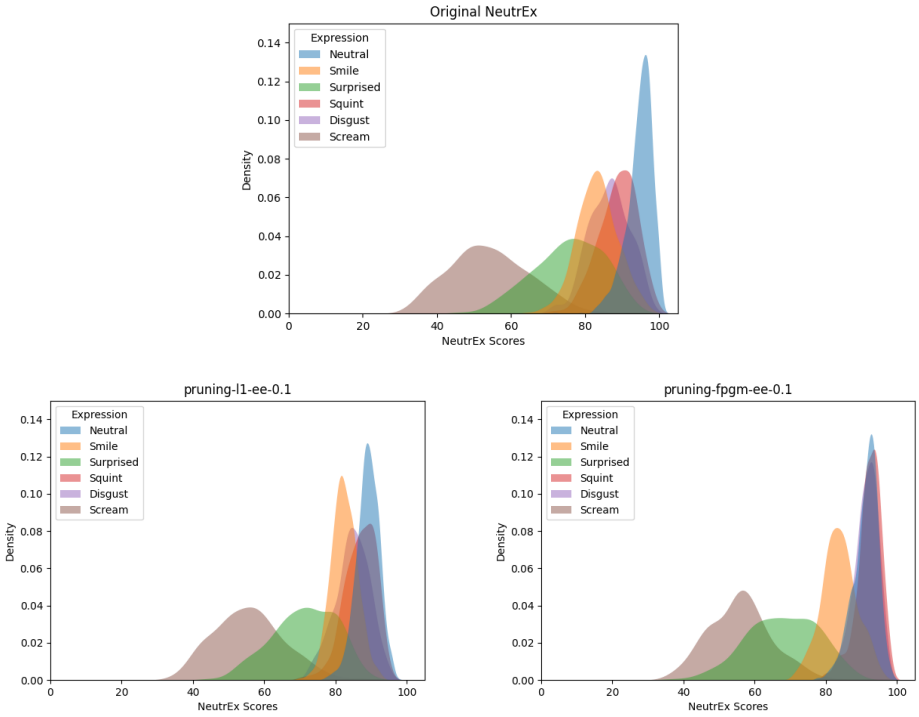

Figure 4.15: Class-wise distribution for `pruning-l1-ee-0.1` (bottom, left) and `pruning-fpgm-ee-0.1` (bottom, right). NeutrEx (top) for comparison. Benchmarked with Multi-PIE.

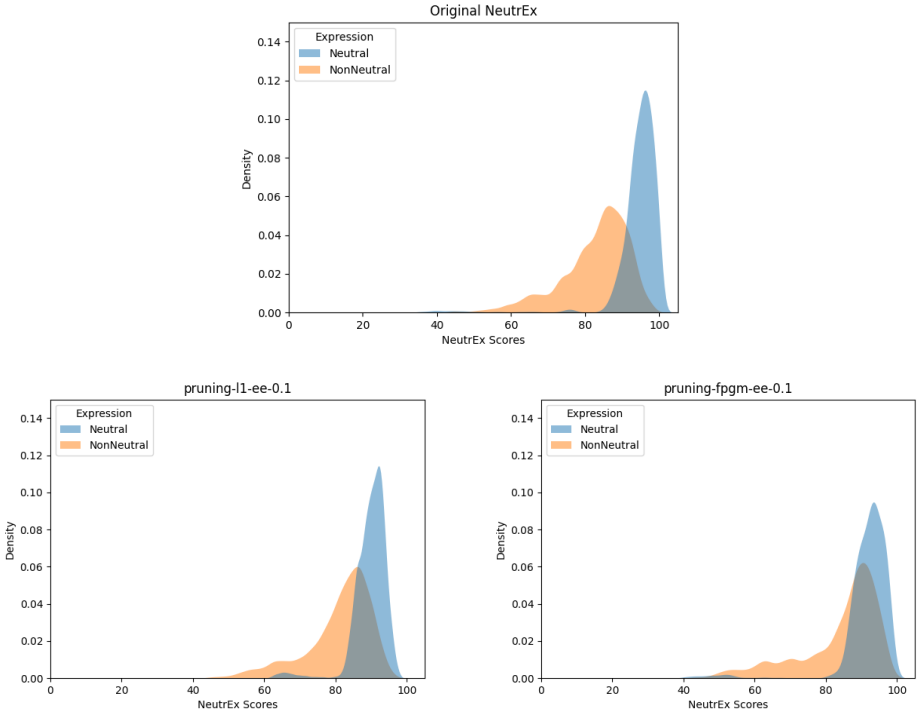

Figure 4.16: Class-wise distribution for `pruning-l1-ee-0.1` (bottom, left) and `pruning-fpgm-ee-0.1` (bottom, right). NeutrEx (top) for comparison. Benchmarked with FEAFA.

Figure 4.16 shows the class-wise distribution where there are only two expressions *Neutral* and *NonNeutral*. We observe more overlap between the two classes in our experiments compared to NeutrEx, especially with the FPGM pruner. Not only that, *Neutral* is also less densely populated for `pruning-fpgm-ee-0.1`, making it less effective for the intended task. On the other hand, `pruning-l1-ee-0.1` performs better, but still has the overlapping issue albeit with an identical distribution for *Neutral* when compared with NeutrEx.

Between the two pruners, L1 Norm pruner performs better than the FPGM one. We investigate our results at higher sparse ratio to confirm this observation. At the lowest sparse ratio of 0.1, pruning the expression encoder is not as effective as pruning the coarse shape encoder. Figure 4.17, 4.18 and 4.19 show our results with sparse ratio 0.3, which also includes the problematic `pruning-l1-ee-0.3`.

Our experiments remain competitive in the EDC curves when benchmarked with Multi-PIE, but perform worse when benchmarked with FEAFA+. For the FEAFA+ dataset, `pruning-l1-ee-0.3` performs better than `pruning-fpgm-ee-0.3` in the pAUC by 0.05%, but outperformed by NeutrEx considerably. Increasing the sparse ratio seems to have an adverse and immediate impact on the performance, despite the promising results with the EDC curves in previous experiments with 0.1 sparse ratio.

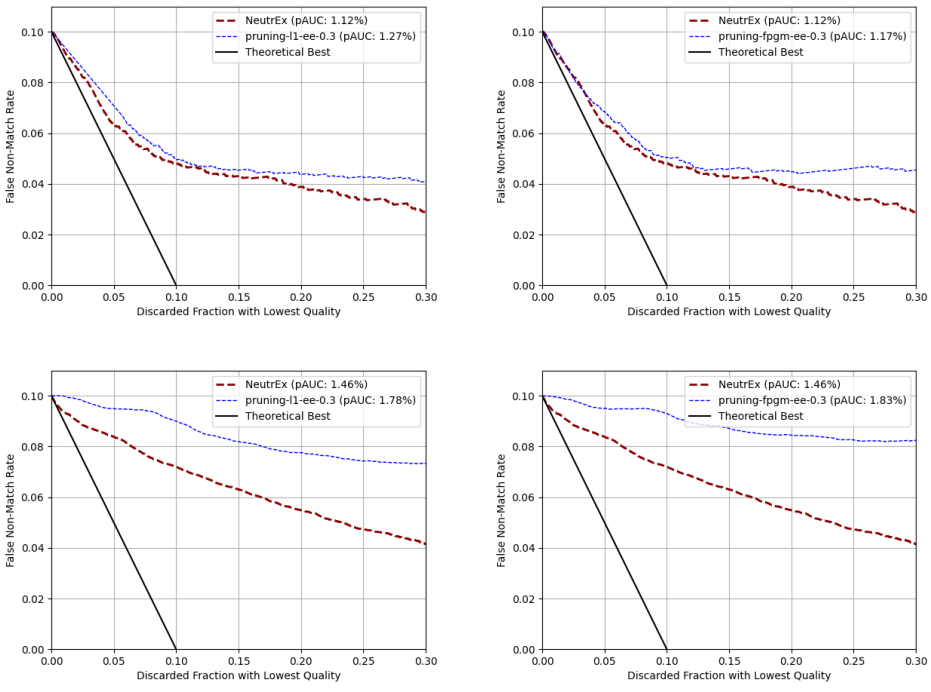

Figure 4.17: EDC curves for `pruning-l1-ee-0.3` (left) and `pruning-fpgm-ee-0.3` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

In terms of class-wise distribution, both experiments fail to deliver results. For the Multi-PIE dataset, `pruning-fpgm-ee-0.3` has extreme overlapping

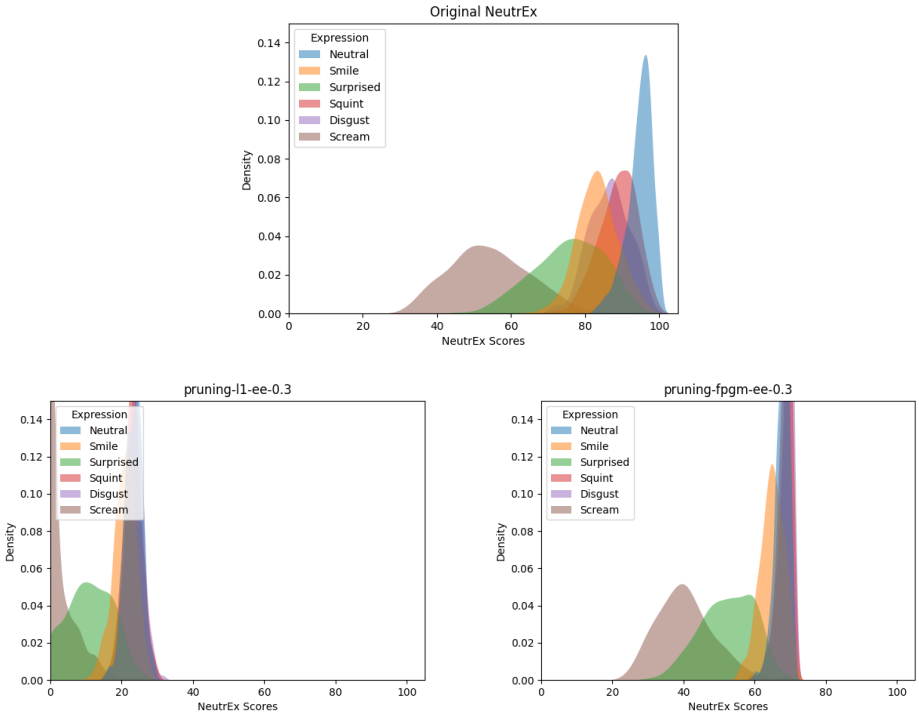

Figure 4.18: Class-wise distribution for `pruning-l1-ee-0.3` (bottom, left) and `pruning-fpgm-ee-0.3` (bottom, right). NeutrEx (top) for comparison. Benchmarked with Multi-PIE.

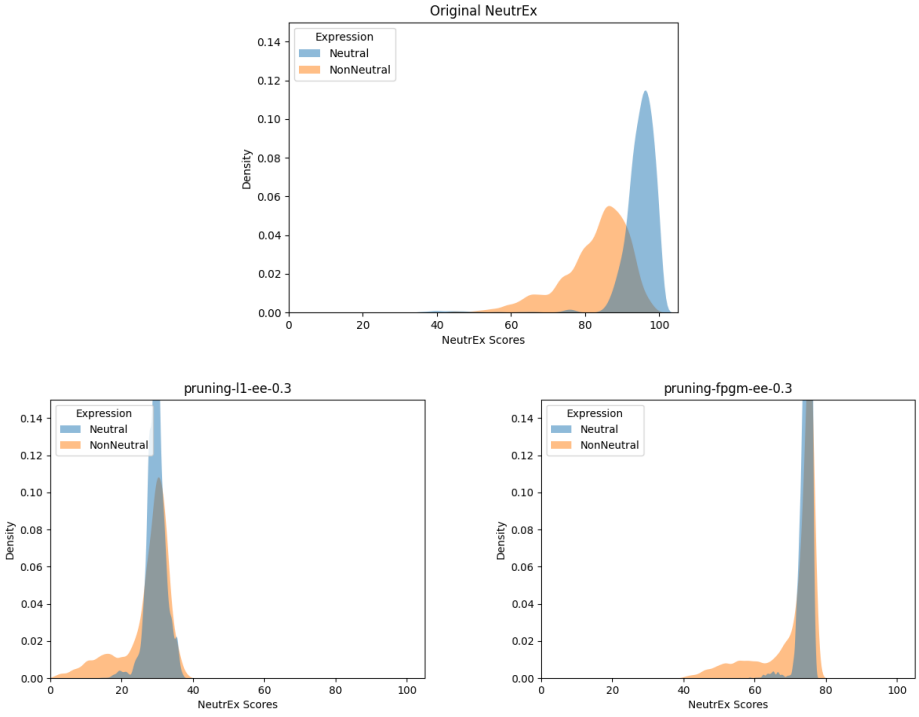

Figure 4.19: Class-wise distribution for `pruning-l1-ee-0.3` (bottom, left) and `pruning-fpgm-ee-0.3` (bottom, right). NeutrEx (top) for comparison. Benchmarked with FEAFA.

problem where *Neutral*, *Squint* and *Disgust* are indistinguishable. Furthermore, this experiment only gives out NeutrEx scores between 20 and 70 out of 100, which goes against the definition of component quality values shall be expressed in the range of [0, 100], where 0 and 100 represent the lower and upper quality boundaries, respectively. `pruning-l1-ee-0.3` perform even worse than `pruning-fpgm-ee-0.3` in this regard, as the scores are only given between 0 and 30. The same observation can be made about their performance with the FEAFA+ dataset, where there is a complete overlap between *Neutral* and *NonNeutral* classes. `pruning-fpgm-ee-0.3` continues to give scores not up to scale, and `pruning-fpgm-ee-0.3` continues to give unusable scores. This in combination with the remark about its parameters count shows that `pruning-l1-ee-0.3` becomes a faulty model after pruning, which implies that pruning can heavily impact the performance if not configured properly.

Experiments with sparse ratio between 0.4 and 0.7 also show similar behaviors in terms of EDC curves and class-wise distribution to `pruning-fpgm-ee-0.3` and `pruning-l1-ee-0.3`. This is consistent with our observation in earlier experiments, i.e after a certain amount of parameters have been removed, the performance gain or loss based on our benchmark is minimal. We conclude our experiments and summarize our findings as follow:

1. The expression encoder to be volatile and more susceptible to changes. At some ratio, the encoder becomes faulty, e.g `pruning-fpgm-ee-0.3`

2. Compared to our success with pruning the coarse shape encoder, our experiments on the expression encoder is considerably worse. The only experiments with acceptable results are at the lowest ratio (0.1), while at higher ratios the results are not suitable for the purpose of NeutrEx w.r.t measuring expression neutrality.

4.4 DISCUSSION

We perform extensive pruning experiments on the coarse shape encoder and the expression encoder. We report our results in Section 4.2 and Section 4.3, respectively. Our pruned coarse shape encoders perform well on our benchmark. At 0.1 sparse ratio, we remove ≈20% of the parameters, while still keeping the performance competitive on all benchmarks. Even at higher ratios which correlate to more parameters removed, our experiments maintain an acceptable performance. Most importantly, all of our experiments maintain distribution and density of the *Neutral* classes in both benchmark datasets, which is crucial to the idea of NeutrEx as a component quality measure.

On the other hand, our experiments with the expression encoder report less promising results. Outside of one experiment at 0.1 sparse ratio (`pruning-l1-ee-0.1`), all other experiments observe the same problem of unable to maintain good class-wise distribution. Note that the experiments still perform to some degree w.r.t EDC curves, which shows the important of com-

paring multiple characteristics to fully understand the performance of NeutrEx and neural networks in general. Some experiments also show unreliable outcomes compared to coarse shape encoder experiments, which is a sign of complication when working on the expression encoder. Furthermore, the overall worse results when we make changes to the expression encoder indicate that for the NeutrEx measure, information from the expression encoder is more valuable to the score than the coarse shape encoder. We conclude that the expression encoder is *a*, volatile and difficult to prune and *b*, more important than the coarse shape encoder. We investigate methods to improve the expression encoder later in this work.

Our experiments shows that from our chosen algorithm: *L1 Norm*, *L2 Norm* and *FPGM*, the choice of which pruner to use does not have a significant impact on the outcome. Instead, our experiments put a heavy emphasis on finding the right sparse ratio to prune the model, and trying to keep the performance to an acceptable level instead of removing as many parameters as possible. Due to the chosen method of pruning where interconnecting components are removed from the model, using a higher sparse ratio can lead to over-pruning the model, leaving only a handful of parameters left in the model. While the reported results are promising, having too few parameters left can be an issue for further operations after pruning, for example quantization or fine-tuning to relearn the information. Our implementation of pruning does not involve re-calibration or retraining the models, as the parameters are not updated, but only removed based on our chosen criterion.

There are multiple ways to extend our pruning experiments. For example, we can implement a different pruning method: Instead of remove parameters at each layer equally, we investigate the importance of them and remove more parameters from the less important ones. Another method is only to maintain information that is needed for NeutrEx: Since there are some parameters in the FLAME [Li+17a] space that are not inherently useful for the NeutrEx measure such as lighting code l or distance to camera c, we can prune components from the encoders so that we no longer have to infer those parameters. Additional experiments can also be conducted with the sparse ratio, in which we increase the ratio with smaller increment to pinpoint the exact sparse ratio for each encoder. All of the examples mentioned are considered during our work, but we are unable to implement those due to the time constraint.

5

FINE-TUNING

We report the results of our fine-tuning experiments in this chapter. We explain the intuition behind fine-tuning strategies in Section 5.1. We fine-tune the expression encoders and report the results in Section 5.2. The overall results are then discussed in Section 5.3.

5.1 INTRODUCTION

Fine-tuning is a transfer learning method, in which a pretrained model acquires new knowledge post-training by learning from additional data. Fine-tuning is used to extend the model's capability to a specific task or domain. In the field of natural language processing, this method is frequently used to apply pretrained large language models to smaller task by [Tou+23; Fab+20; Zha+20; Dod+20]. Although not as extensively researched, fine-tuning is also a viable option in computer vision tasks [Mah+19; Too+19; Guo+19]. One important advantage of fine-tuning is reducing the requirements for dataset and hardware during training, thus allowing changes to be made to without completely retraining the model from scratch. In the context of reducing parameters from neural network, fine-tuning has been successfully included as complement for other methods [Han+15], or as the sole method of parameters reduction [Din+23].

As remarked in the closing section of Chapter 4, our pruned encoders are not inherently guided to a specific task, but rather a collection of parameters remained after removing unimportant ones. By applying fine-tuning, we can train our pruned encoders to be more suitable to our task, which is to contribute to the expression neutrality measure NeutrEx. Not only that, this also gives us the opportunity to improve our performance to be as closed to the original NeutrEx as possible. We propose a fine-tuning schema in Figure 5.1.

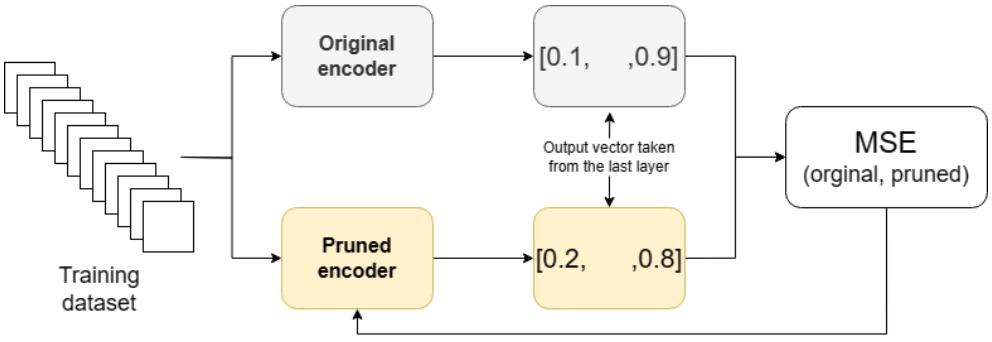

Figure 5.1: Fine-tuning schema for pruned encoders. Values in the output vectors are for illustration only.

Our fine-tuning schema is similar to neural network training, in which information is learned from a dataset. We incorporate the original encoders of NeutrEx into our fine-tuning. We infer each image in the training dataset twice, using the pruned encoder and the corresponding original encoder, i.e if we are fine-tuning a pruned coarse shape encoder, we also run the image through the original coarse shape encoder. From each encoder, we receive the output as a vector of floating numbers. We calculate the mean squared error between the two vectors, and use this information to optimize the parameters in the pruned encoder. This is in essence a *knowledge distillation* setup; in which the original encoder serves as the teacher model, and directly teaches the student model to match its output.

We perform fine-tuning on the pruned encoders from the experiments in Chapter 4. Since the encoders in this case already have a fixed structure with fewer parameters after pruning, we do not have to define an entirely new structure to solve the original task of removing parameters in NeutrEx. The coarse shape encoders already perform well according to our observation, so we focus mainly on fine-tuning the pruned expression encoders in the following reports.

For training and validation, we use our END collection and AffectNet [MHM17]. In experiments fine-tuned with END, we discard all images from FEAFA+ and Multi-PIE, since they are used later in the benchmarking. This leaves us with ≈3000 images in total for training and validation. In experiments fine-tuned with AffectNet, due to time constraints and hardware limitation, we use the first 30k images out of 400k images in the dataset for training, and 3k subsequent images for validation. There are two main reasons for fine-tuning with two different datasets. First, EMOCA is originally trained with AffectNet. With these two datasets, we can see the effectiveness between fine-tuning with the same dataset and fine-tuning with a different dataset. Second, since AffectNet has considerably more training images and subjects, we can observe the impact of the amount of data available on fine-tuning. We train the pruned encoders in 10 epochs for all fine-tuning experiments.

As with pruning experiments, we refer to our fine-tuning experiments in later discussions in the following format. Note that the combination of three indicators at the end of the notation corresponds with the experiments from previous chapter, where we directly take the encoders from.:

`finetuning-`*`dataset-pruner-encoder-sparse_ratio`*

where

- **dataset** can be `end` for the END collection or `affectnet` for the AffectNet dataset.
- **pruner** can be `l1` for the L1 Norm pruner or `fpgm` for the FPGM pruner.
- **encoder** can be `cse` for the coarse shape encoder or `ee` for the expression encoder. In the scope of this work it will mostly be `ee`.
- **sparse_ratio** can be a floating number between 0.0 and 1.0.

5.2 FINE-TUNING THE EXPRESSION ENCODER

We decide to fine-tune only a subset of pruned expression encoders, instead of all of them. We believe that fine-tuning encoders with more parameters remain is more sensible to fine-tuning ones with fewer parameters, as the training is more likely to converge if there are more parameters to optimize. Out of the two pruners available in our pruning expression encoder experiment (Section 4.3), we focus on the L1 Norm variant, as the results from this variant are slightly more promising than the alternative. Based on this reasoning, we choose the pruned encoders `pruning-l1-ee-0.2` and `pruning-l1-ee-0.3` to fine-tune. The 0.1 variant is left out, as the performance is acceptable after pruning. Note that we include the faulty `pruning-l1-ee-0.3`, as an attempt to improve the performance by fine-tuning.

First, we fine-tune on the END collection. Figure 5.2, 5.3, 5.4 and 5.5 show the benchmarking performance for `finetuning-end-l1-ee-0.2` and `finetuning-end-l1-ee-0.3`, as well as comparison to the pre-fine-tuning pruned versions `pruning-l1-ee-0.2` and `pruning-l1-ee-0.3`. Note that we do not report number of parameters in this Section, since they remain the same after fine-tuning.

We receive worse results with the EDC curves (Figure 5.2 and 5.3) compared to the original NeutrEx. With the Multi-PIE benchmark, the 0.2 variant shows an increase of 0.11% in pAUC after fine-tuning, while the 0.3 has a slight decrease of 0.06 %. In the FEAFA+ benchmark, both fine-tuned encoders perform significantly worse, as the false non-match rate does not decrease even with the increase in discarded fraction. This points to a decrease in performance after fine-tuning in this particular characteristic.

On the contrary, our fine-tuned encoders show significant improvements in class-wise distribution after fine-tuning. With the Multi-PIE dataset (Figure 5.4), `finetuning-end-l1-ee-0.2` is able to realign the score distribution to a new one that strongly resembles the distribution of NeutrEx. The label *Neutral* is now distributed between [80, 100] with a high density, which is sufficient to measure expression neutrality. Expressions that are distant from *Neutral* such as *Surprised* and *Scream* continue to receive lower score. On the other hand, the overlapping problem still persists, as *Squint* and *Disgust* still score very highly unlike in NeutrEx. `finetuning-end-l1-ee-0.3`, however, manages to improve on this aspect: Not only are the scores for those two classes lower in this case, their density is also reduced to a level that resembles the distribution of NeutrEx. This is our first experiment with the expression encoder that can disentangle *Neutral* from other classes, which is encouraging.

Similar improvements can also be observed with class-wise distribution on the FEAFA+ dataset (Figure 5.5). Both experiments improve when compared to the pre-fine-tuned performance, and most crucially also avoids overlapping between *Neutral* and *NonNeutral* classes. We conclude that our experiments succeed w.r.t to the class-distribution characteristic. Experiment `finetuning-end-l1-ee-0.3` recovers from the disappointing results after prun-

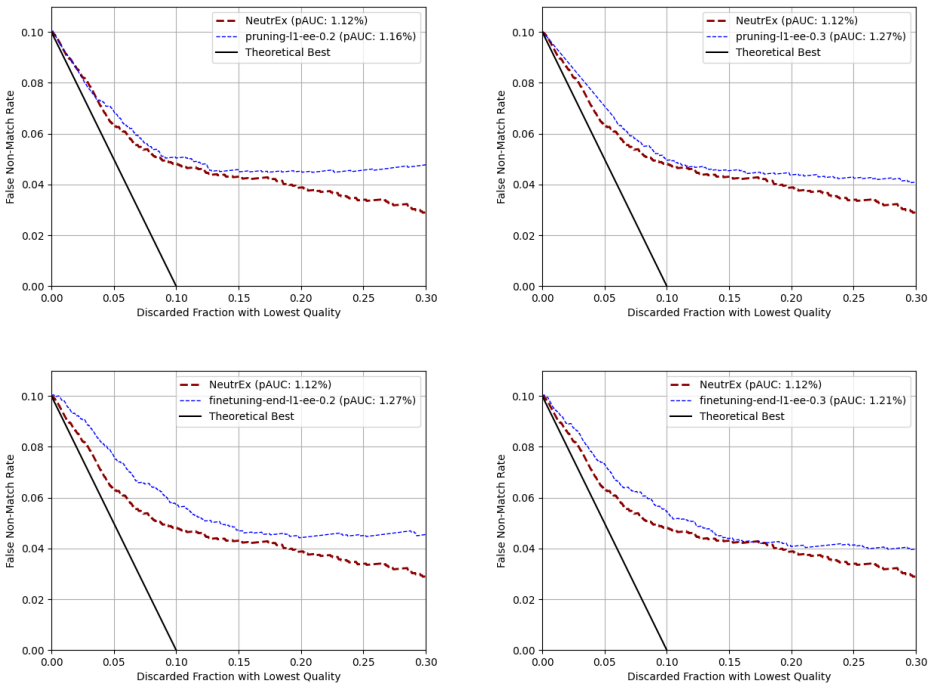

Figure 5.2: EDC curves for `finetuning-end-l1-ee-0.2` (bottom, left) and `finetuning-end-l1-ee-0.3` (bottom, right). `pruning-l1-ee-0.2` (top, left) and `pruning-l1-ee-0.3` (top, right) for comparison. Benchmarked with Multi-PIE.

ing and even outperforms its peer, which shows the effectiveness of fine-tuning when combined with pruning.

Due to the poor performances on the EDC curves, we experiment more with fine-tuning with the goal to improve this aspect. We fine-tune the pruned encoders with the AffectNet dataset [MHM17], which has more data available but less overall quality for training than our END collection. We compare the EDC curves after fine-tuning with AffectNet in Figure 5.6 and 5.7. With the Multi-PIE benchmark, we improve on the previous fine-tuning experiments significantly. Not only that, we outperform NeutrEx by a small margin of 0.06% and 0.05% pAUC. With the FEAFA+ benchmark, we also outperform both the pruned versions and the previously fine-tuned versions; but still fail to reach NeutrEx's performance. Nevertheless, this is an overall improvement, which can be directly attributed to changing the fine-tuning dataset.

The class-wise distributions are compared in Figure 5.8 and 5.9. As with our report on the EDC curves, fine-tuning with AffectNet also improves our performance on this characteristic. `finetuning-affectnet-l1-ee-0.2` shows a clearer distribution compared to its END counterpart in the Multi-PIE benchmark, and also avoids the overlapping problem with the *Neutral* class. `finetuning-affectnet-l1-ee-0.3` improves further on its result by disentangling *Neutral* from *Squint* and *Disgust* almost completely. Including NeutrEx, this is the first time *Neutral* has been separated significantly from

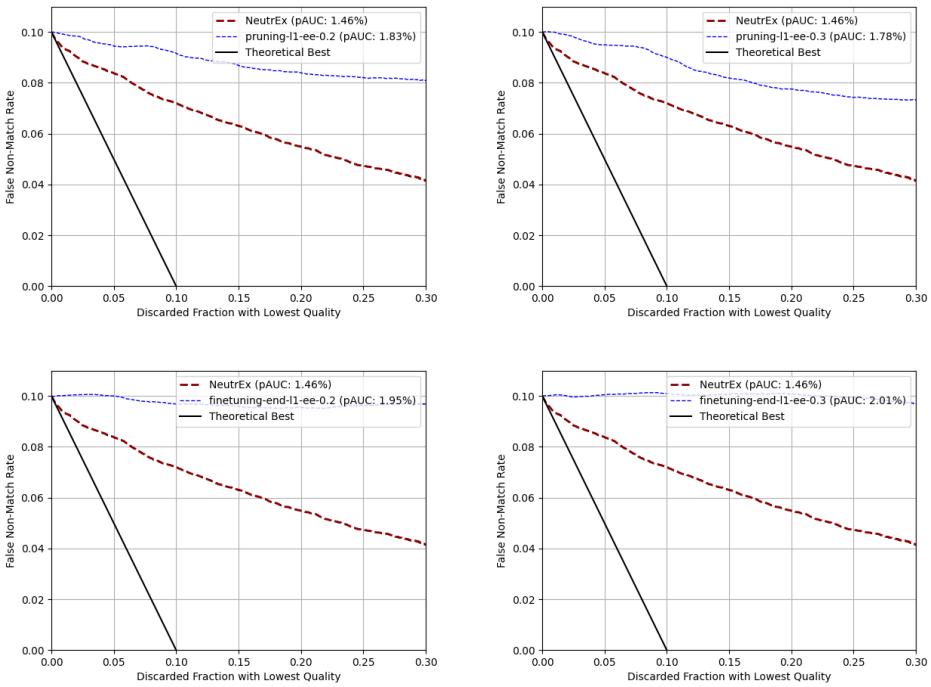

Figure 5.3: EDC curves for `finetuning-end-l1-ee-0.2` (bottom, left) and `finetuning-end-l1-ee-0.3` (bottom, right). `pruning-l1-ee-0.2` (top, left) and `pruning-l1-ee-0.3` (top, right) for comparison. Benchmarked with FEAFA+.

other labels in this dataset. Both experiments give lower scores to *Scream* compared to NeutrEx, which we see as an improvement since it is the most distant expression compared to *Neutral*. On the FEAFA+ benchmark, we only observe some slight changes compared to previous experiments.

Due to the time constraints, we only manage to perform fine-tuning experiments on `pruning-l1-ee-0.2` and `pruning-l1-ee-0.3`. We conclude our experiments and report our finding as follow:

1. Fine-tuning improves our pruned expression encoder significantly.
2. The performance after fine-tuning is comparable to original NeutrEx.
3. Fine-tuning with a larger dataset yields better result in this instance.

5.3 DISCUSSION

We perform fine-tuning on a subset of pruned expression encoders in this Chapter. While we are not able to perform on all encoders from Section 4.3, our current experiments show promising results. Not only the performance improves significantly compared to the pruned ones, it is also comparable to the original NeutrEx on some characteristics. Since fine-tuning introduces task-specific guidance to the model, it is the logical follow-up method after pruning. Our experiments show that fine-tuning can be used to fix severe

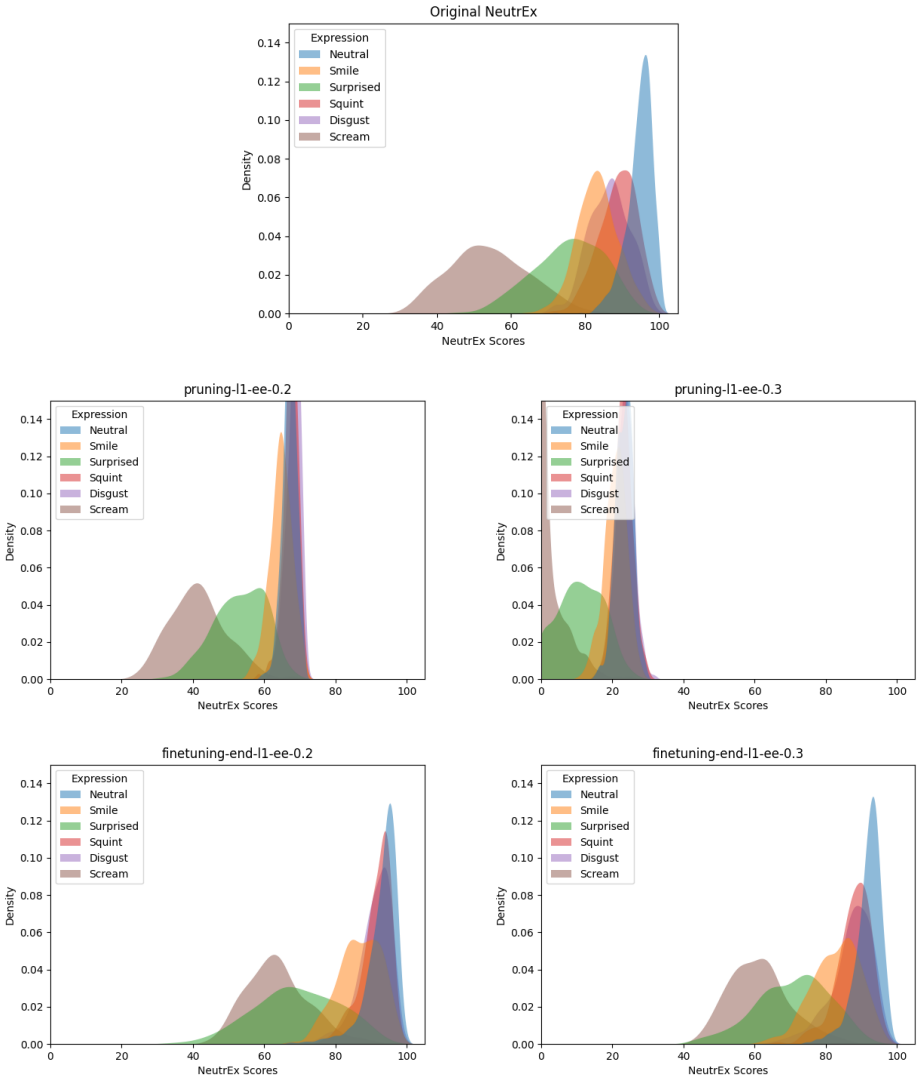

Figure 5.4: Class-wise distribution for `finetuning-end-l1-ee-0.2` (bottom, left) and `finetuning-end-l1-ee-0.3` (bottom, right). NeutrEx (top), `pruning-l1-ee-0.2` (middle, left) and `pruning-l1-ee-0.3` (middle, right) for comparison. Benchmarked with Multi-PIE.

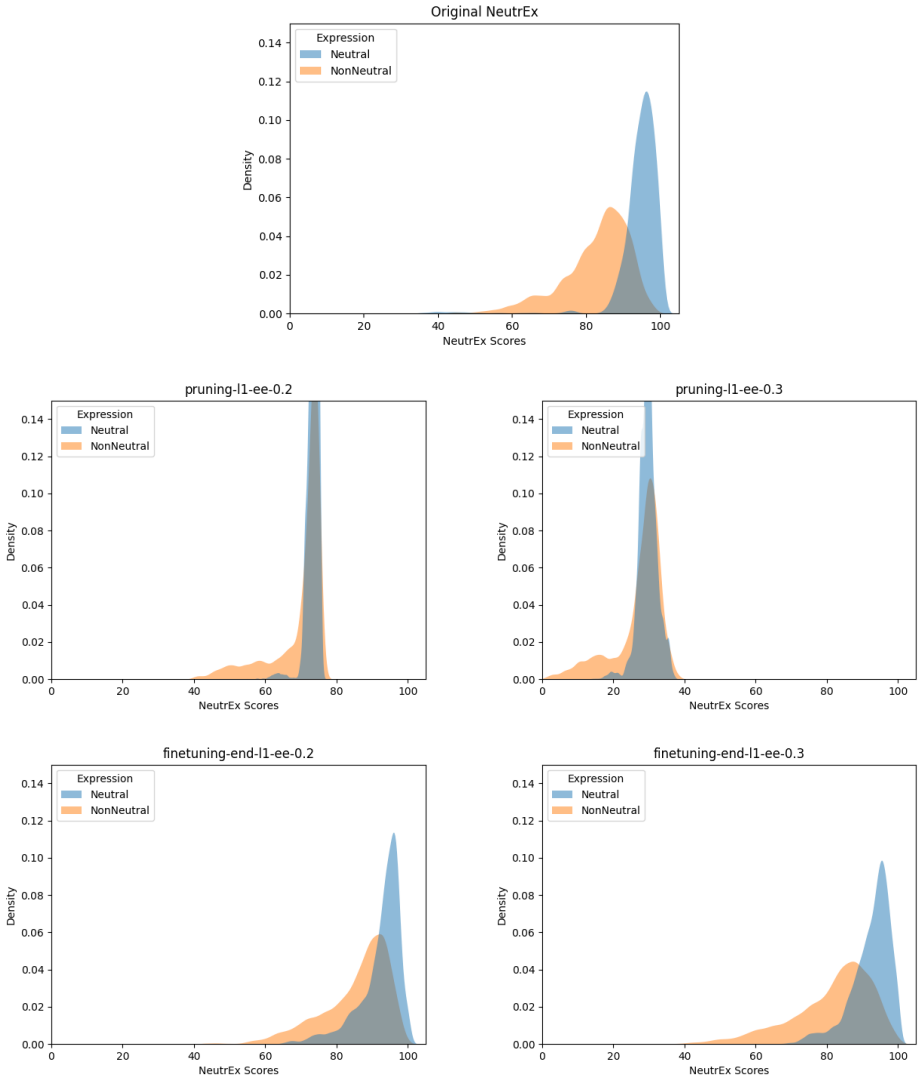

Figure 5.5: Class-wise distribution for `finetuning-end-l1-ee-0.2` (bottom, left) and `finetuning-end-l1-ee-0.3` (bottom, right). NeutrEx (top), `pruning-l1-ee-0.2` (middle, left) and `pruning-l1-ee-0.3` (middle, right) for comparison. Benchmarked with FEAFA+.

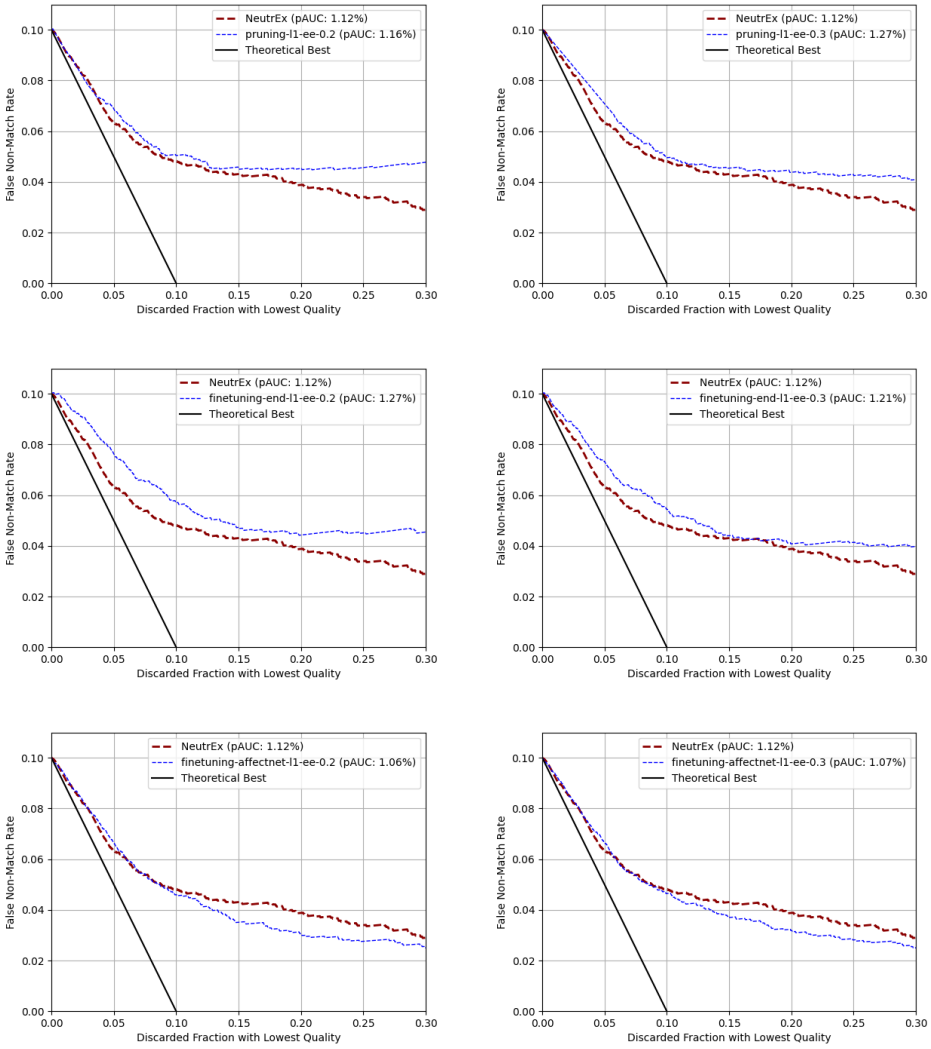

Figure 5.6: EDC curves for `finetuning-affectnet-l1-ee-0.2` (bottom, left) and `finetuning-affectnet-l1-ee-0.3` (bottom, right). `pruning-l1-ee-0.2` (top, left) and `pruning-l1-ee-0.3` (top, right), `finetuning-end-l1-ee-0.2` (middle, left) and `finetuning-end-l1-ee-0.3` (middle, right) for comparison. Benchmarked with Multi-PIE.

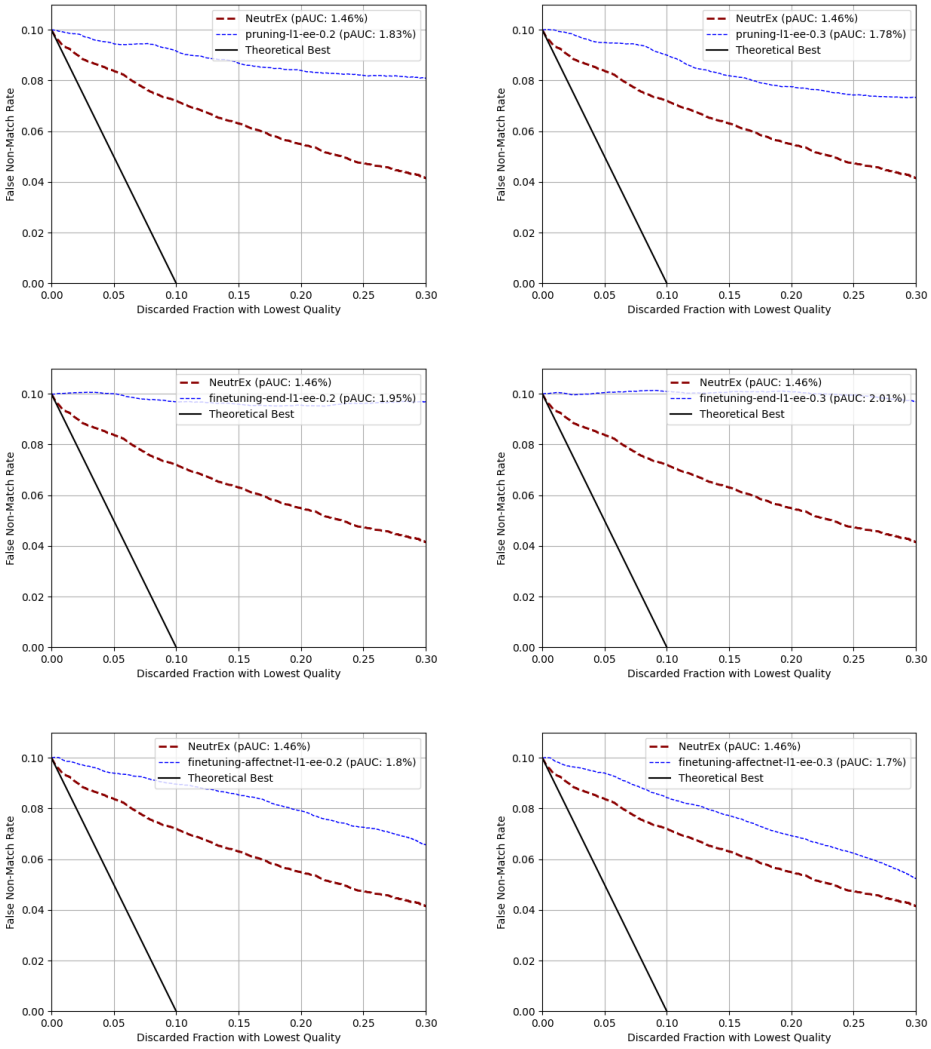

Figure 5.7: EDC curves for `finetuning-affectnet-l1-ee-0.2` (bottom, left) and `finetuning-affectnet-l1-ee-0.3` (bottom, right). `pruning-l1-ee-0.2` (top, left) and `pruning-l1-ee-0.3` (top, right), `finetuning-end-l1-ee-0.2` (middle, left) and `finetuning-end-l1-ee-0.3` (middle, right) for comparison. Benchmarked with FEAFA+.

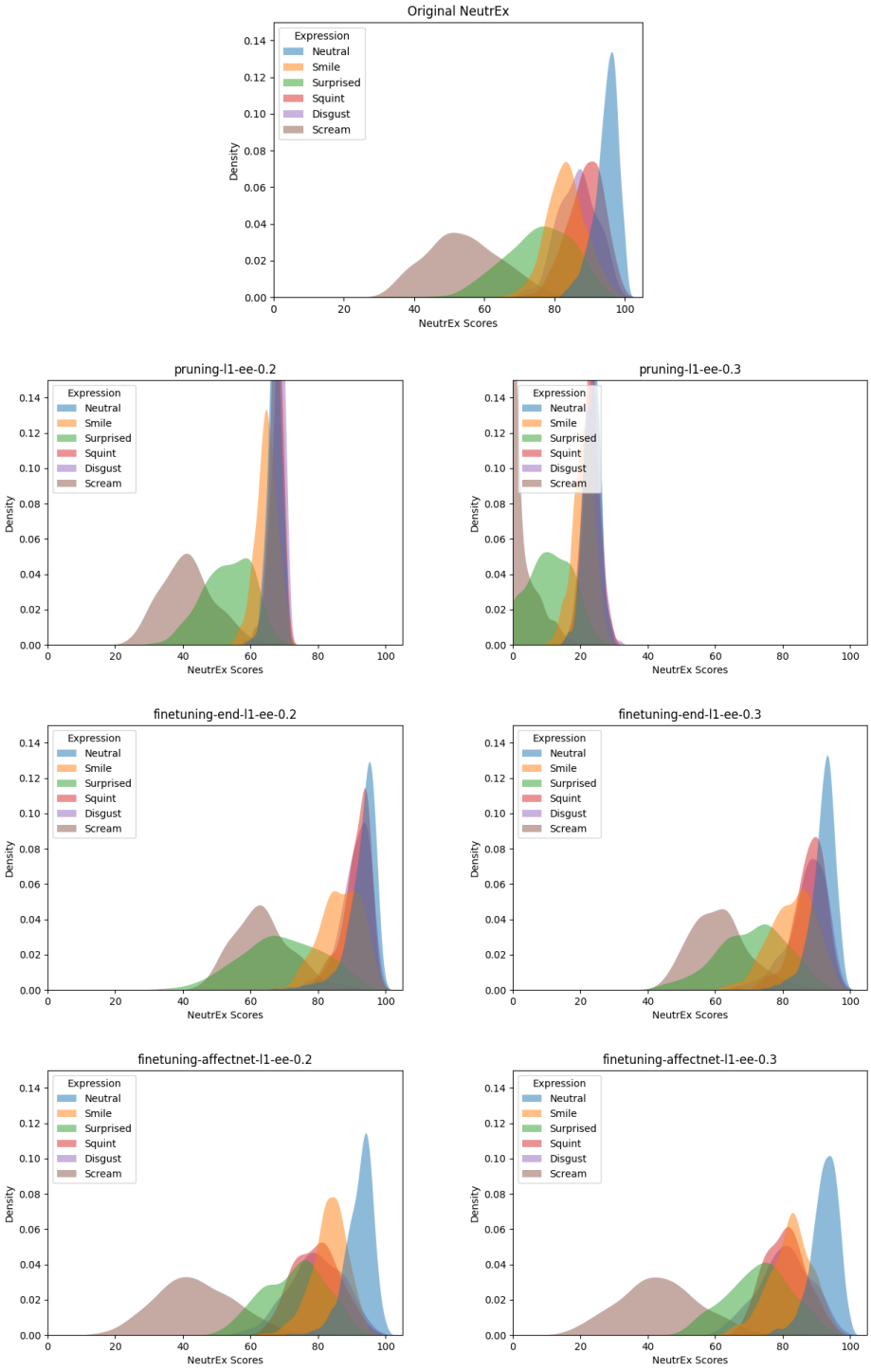

Figure 5.8: Class-wise distribution for `finetuning-affectnet-l1-ee-0.2` (bottom, left) and `finetuning-affectnet-l1-ee-0.3` (bottom, right). NeutrEx (top), `pruning-l1-ee-0.2` (2nd row, left) and `pruning-l1-ee-0.3` (2nd row, right), `finetuning-end-l1-ee-0.2` (3rd row, left) and `finetuning-end-l1-ee-0.3` (3rd row, right) for comparison. Benchmarked with Multi-PIE.

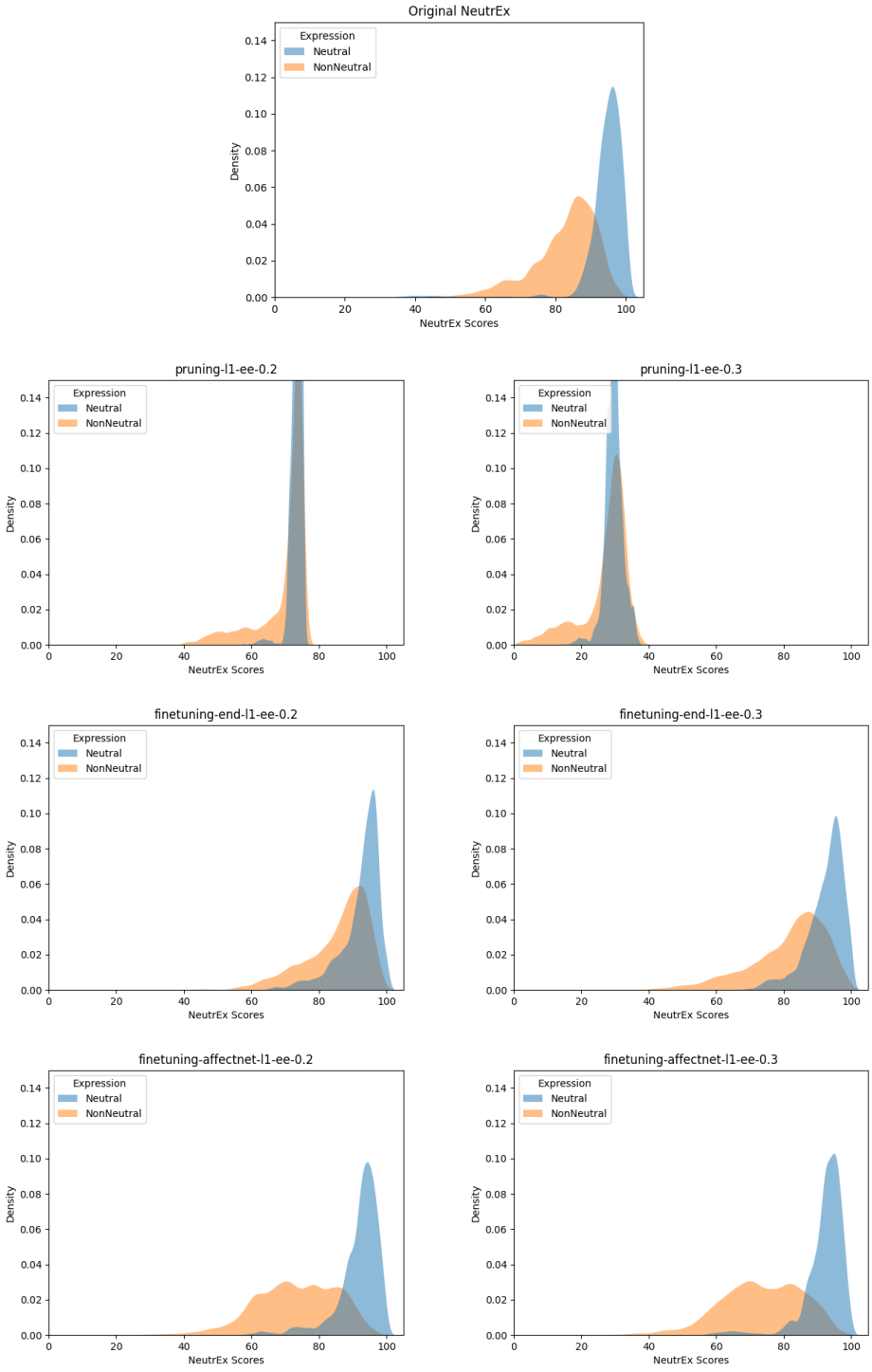

Figure 5.9: Class-wise distribution for `finetuning-affectnet-l1-ee-0.2` (bottom, left) and `finetuning-affectnet-l1-ee-0.3` (bottom, right). NeutrEx (top), `pruning-l1-ee-0.2` (2nd row, left) and `pruning-l1-ee-0.3` (2nd row, right), `finetuning-end-l1-ee-0.2` (3rd row, left) and `finetuning-end-l1-ee-0.3` (3rd row, right) for comparison. Benchmarked with FEAFA+.

performance degradation after pruning, as shown in `finetuning-end-l1-ee-0.3` and `finetuning-affectnet-l1-ee-0.3`. We find this observation important, as it shifts the performance-related concerns to the fine-tuning step, and allows the pruning step to focus solely on reducing as much parameters as possible. As a caution, based on some of our failed experiments which we do not report in earlier discussion, having too few parameters makes the training during fine-tuning unable to converge as seen in Figure 5.10.

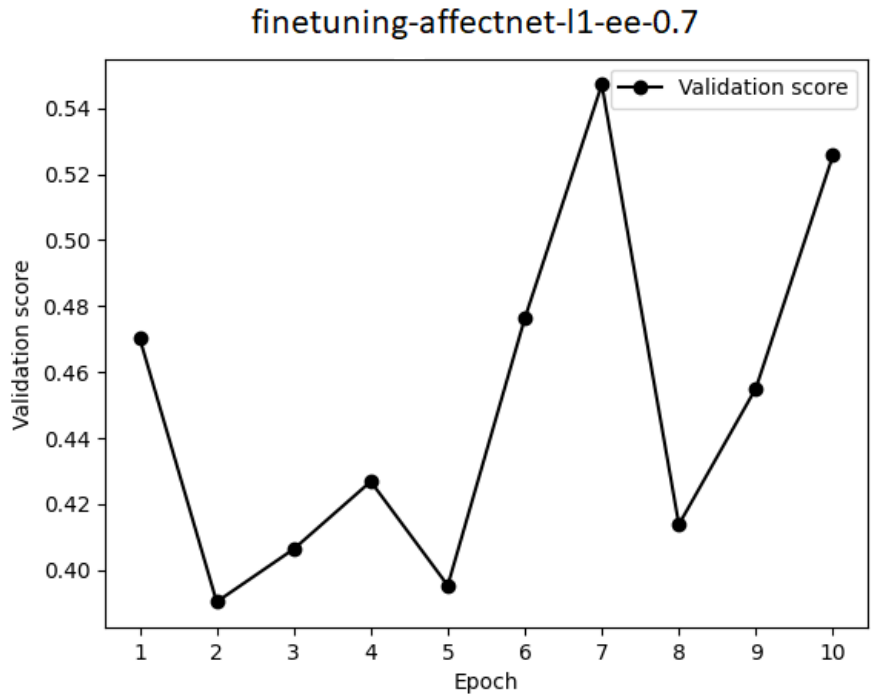

Figure 5.10: Validation loss of `finetuning-affectnet-l1-ee-0.7` during training.

In our limited scope of experiments, the choice of fine-tuning dataset has an effect on the outcome. Our END collection has considerably less images, but they are of higher quality: All images are taken in controlled environment, with adequate lighting and the subjects face the camera directly. This is aligned with how NeutrEx are envisioned, as the authors of NeutrEx also normalize camera distance and lighting during the score computation. On the other hand, AffectNet has more images but of lower quality, as they are taken in-the-wild and not controlled. Some of them are not usable in our training process, since the built-in face recognition of the encoder does not recognize facial details from these images. Nevertheless, our fine-tuning with AffectNet yields better results, which can be anecdotally attributed to the higher number of images. Given that we only use a subset of AffectNet for training, it is also possible for us to extend the scope of training and fine-tune our encoders with more data to make a comprehensive observation on this topic in later works.

In the context of NeutrEx, we can also apply fine-tuning to the pruned coarse shape encoders, which currently report satisfying results after prun-

ing only. Since fine-tuning already improves our pruned expression encoders substantially, it is reasonable to expect that it can also improve our coarse shape encoders. We can also improve our fine-tuning method to be more aligned with how NeutrEx are calculated, by giving more weights during training to parameters that are used in NeutrEx such as head shape code β or pose code θ. Since our method is based on knowledge distillation, it is also a viable direction to define a completely new network with a suitable architecture that has fewer parameters, and use our fine-tuning method to teach the new model information from the encoders.

6

NEUTREX-LITE

After experimenting with the encoders, we complete NeutrEx-lite by substituting both the coarse shape encoder and the expression encoder in NeutrEx with our smaller versions. Since our encoders are designed to have the same inputs and outputs as NeutrEx's encoders, we can replace them directly without making changes to other components.

There are two main focuses for our experiments: To remove the most parameters from the original encoder, and to maintain the performance as close as possible to NeutrEx. Our experiments show that it is not possible for a singular encoder to reach both targets equally, i.e the experiments that remove more parameters have worse performance and vice versa. Furthermore, due to the large number of experiments on both encoders, it is time-consuming to test all possible combinations to determine the best implementation for NeutrEx-lite. In the scope of this work, we limit our testing to a few versions of NeutrEx-lite, which are constructed from a small subset of experiments. This subset of experiments contains two coarse shape encoders and two expression encoders. Among the coarse shape encoders we have available, we choose the one with the one with **best performance** and the one with the **least parameters**, which represent the two aspects of our focus. We choose two expression encoders with the same reasoning. Our subset of experiments contains the following experiments which we have extensively discussed in previous Chapters.

1. Coarse shape encoder with best performance: `pruning-l2-cse-0.1`. Even though `pruning-fpgm-cse-0.1` slightly outperforms this experiment in terms of pAUC, it has the closer class-wise distribution compared to NeutrEx. We believe the latter is more important, and choose `pruning-l2-cse-0.1` between the two.

2. Coarse shape encoder with least parameters: `pruning-l2-cse-0.7`.

3. Expression encoder with best performance: `finetuning-affectnet-l1-ee-0.3`.

4. Expression encoder with least parameters: `pruning-l1-ee-0.7`.

We create pairs of encoders from the chosen ones by combining one coarse shape encoder and one expression encoder each. This results in four versions of NeutrEx-lite as seen in Table 6.1. We also report the number of parameters in Table 6.2.

Out of all versions, `neutrex-lite-a` has fewest parameters, but the model is no longer functional. Figure 6.1 shows the EDC curves for this experiment. Its performance worsens significantly in the Multi-PIE when more images

Name	Coarse shape encoder		Expression encoder	
	Best Performance	Least Parameters	Best Performance	Least Parameters
	1	2	3	4
`neutrex-lite-a`		✓		✓
`neutrex-lite-b`		✓	✓	
`neutrex-lite-c`	✓			✓
`neutrex-lite-d`	✓		✓	

Table 6.1: Our versions of NeutrEx-lite. Checkmark ✓ indicates the corresponding encoder is used.

Name	**#. of parameters** (CSE)	**#. of parameters** (EE)	**Total #. of parameters**	**Total size on storage (MB)**
`neutrex-lite-a`	26,956	21,748	48,704	11.44
`neutrex-lite-b`	26,956	12,643,219	12,670,175	181.79
`neutrex-lite-c`	21,011,447	21,748	21,033,195	250.89
`neutrex-lite-d`	21,011,447	12,643,219	33,654,666	421.24

Table 6.2: Pruning the coarse shape encoder with L1 Norm pruner

are discarded, although for FEAFA+ the false non-match rate remains comparable to other experiments in previous chapters. A closer look at the outputs of the model reveals that it measures all images with nearly identical scores, which indicates a pattern of over-generalization. We conclude that choosing the most efficient encoders parameter-wise is not a viable option for NeutrEx-lite.

By replacing the expression encoder from `neutrex-lite-a with` the best performance one, `neutrex-lite-b` performs well on all of our benchmarks. As reported in Figure 6.2, this version has a pAUC of 1.1% compared to 1.2% of NeutrEx when benchmarked with Multi-PIE, but 0.28% higher pAUC with FEAFA+. `neutrex-lite-b`'s class-wise distributions closely resembles those of NeutrEx in both datasets, with the exception of *Surprised* in Multi-PIE.

On the other hand, replacing the coarse shape encoder with the best performance one does not recover the performance. `neutrex-lite-c` both under-performs on both EDC benchmarks (Figure 6.4) by 0.07% and 0.36% respectively. Not only that, we once again observe the severe overlapping problem in both Multi-PIE and FEAFA+ (Figure 6.5), which is the main issue that we have with most expression encoder experiments before fine-tuning. Like `neutrex-lite-a`, this version is also not a viable combination for NeutrEx-lite.

Version `neutrex-lite-d` retains the most parameters among our four versions. Since we focus on performance for both encoders for this version,

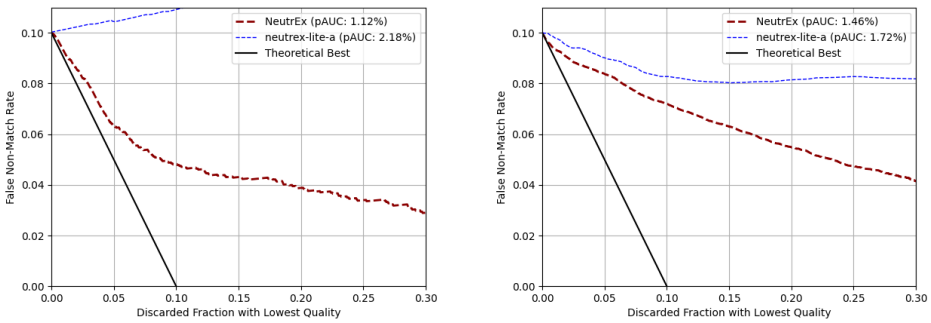

Figure 6.1: EDC curves of `neutrex-lite-a`. Benchmarked with Multi-PIE (left) and FEAFA+ (right).

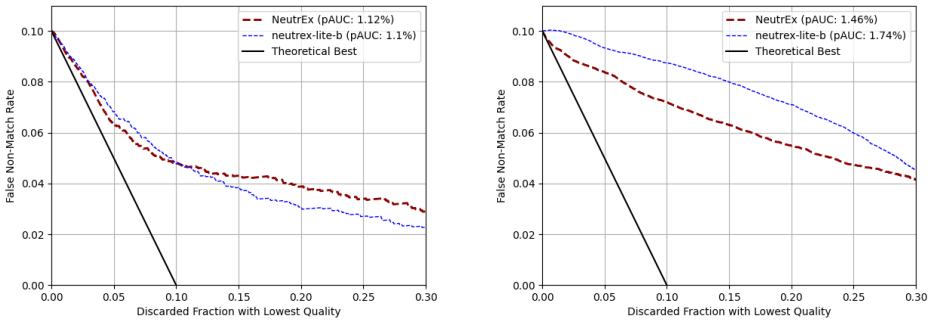

Figure 6.2: EDC curves of `neutrex-lite-b`. Benchmarked with Multi-PIE (left) and FEAFA+ (right).

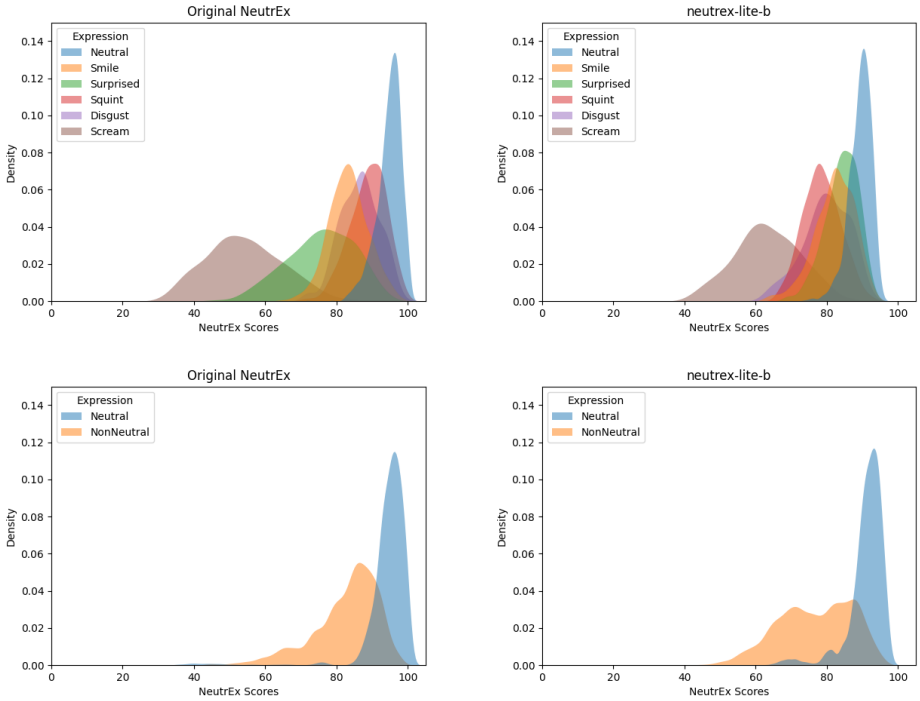

Figure 6.3: Class-wise distributions comparison between NeutrEx (left) and `neutrex-lite-b` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

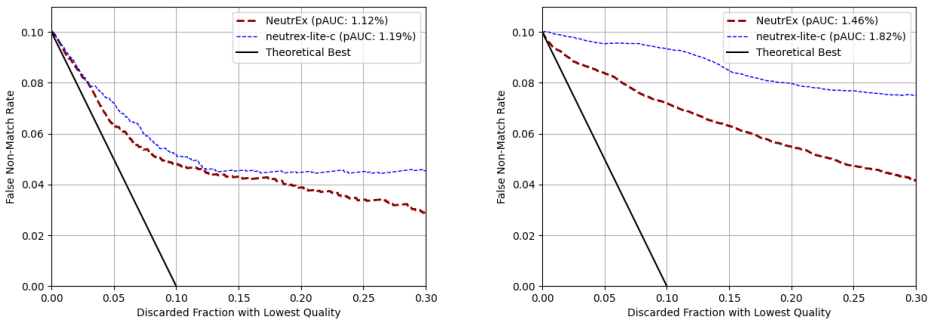

Figure 6.4: EDC curves of `neutrex-lite-c`. Benchmarked with Multi-PIE (left) and FEAFA+ (right).

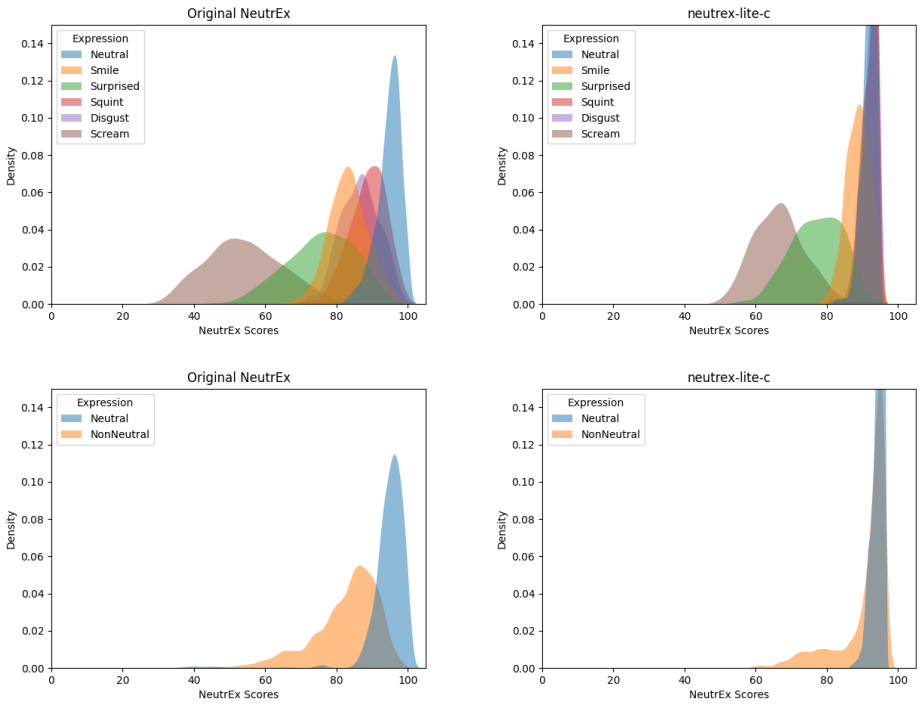

Figure 6.5: Class-wise distributions comparison between NeutrEx (left) and `neutrex-lite-c` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

we expect a higher overall performance compared to previous ones. Figure 6.6 shows that in terms of EDC curves, we slightly outperform NeutrEx in Multi-PIE. In FEAFA+ we have a higher pAUC, despite both encoders report more competitive values in their individual experiments. Our class-wise distributions in Figure 6.7 also show some minor performance degradation, but remain competitive with NeutrEx. Most importantly, the *Neutral* classes shows no overlapping with other labels.

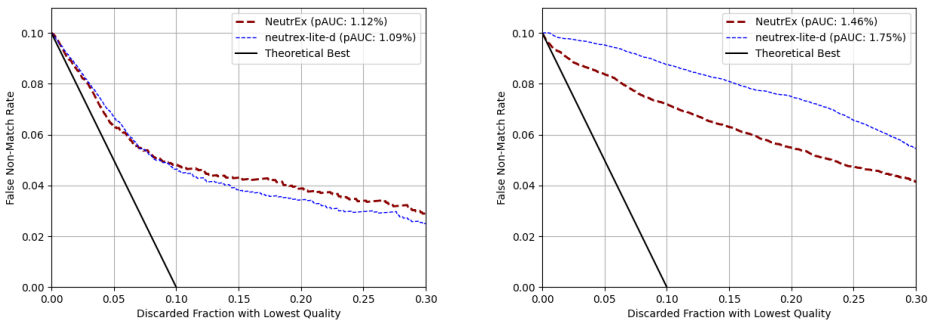

Figure 6.6: EDC curves of `neutrex-lite-d`. Benchmarked with Multi-PIE (left) and FEAFA+ (right).

Among the four proposed versions, `neutrex-lite-b` and `neutrex-lite-d` are the standout versions. We show that there are little differences between

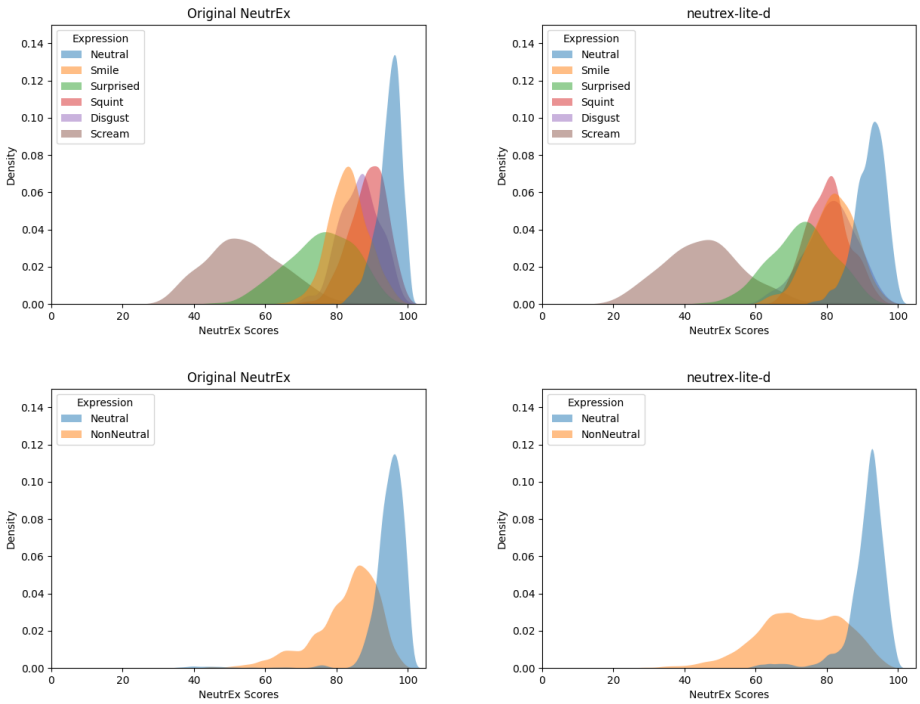

Figure 6.7: Class-wise distributions comparison between NeutrEx (left) and `neutrex-lite-d` (right). Benchmarked with Multi-PIE (top) and FEAFA+ (bottom).

these two versions in all of our benchmark. Despite having the better coarse shape encoder performance-wise, `neutrex-lite-d` fails to outperform `neutrex-lite-b` decisively, which is surprising. Since they both use the best performance expression encoder, this is a possible indication that the expression encoder is more impactful to the NeutrEx measure than the coarse shape encoder. This is consistent with our previous assessment, which is making changes to the expression encoder causes noticeable performance degradation. Parameter-wise, the `neutrex-lite-b` version has $\approx$11 million parameters fewer than `neutrex-lite-d`, and $\approx$38 million parameters fewer than the original NeutrEx.

Overall, the results from combining smaller versions of the coarse shape encoder and the expression encoder are within our expectations. We show that it is possible to combine our previous researches and still maintain good performance, albeit with some slight degradation. We proposes `neutrex-lite-b` as the ideal version of NeutrEx-lite both performance- and parameter-wise, and conclude our report on experiments.

7 CONCLUSION

7.1 DISCUSSION

Our main focus is to create a more efficient version of NeutrEx in terms of computational overhead, while still maintaining its effectiveness as a quality measure based on the current draft international standard of ISO/IEC 29794-5 [ISO23b]. This version of NeutrEx - named **NeutrEx-lite** - is implemented by applying well-researched methods in the field of neural network optimization. In our experiments, we show that these methods are effective for our task. Our final result has considerably fewer parameters than the original NeutrEx (≈38 million parameters fewer), and still performs competitively with NeutrEx on our benchmarks. We summarize our findings and results as follow.

In the scope of this work, we experiment mainly with *Pruning* and *Knowledge distillation*. Our method of pruning removes parameters structurally from the network: First, parameters are ranked based on their importance to the inference capability of the model. The least important ones are removed from the network, and also all the parameters that are interconnected with those. Our experiments show that this method is capable of removing a large number of parameters from the network while still retains performance to a degree. For example, our pruned expression encoders perform closely to NeutrEx, even with ≈20% of the parameters removed. There are two main factors that can affect the effectiveness of pruning, namely the *sparse ratio*, i.e how may parameters should be removed and the *pruning algorithm*, i.e how the importance of parameters should be derived. We put an emphasis on finding the right sparse ratio, rather than finding the right algorithm in our work.

However, our experiments with the expression encoder also show that pruning alone is not enough for the results to be competitive with NeutrEx. We observe that pruning the expression encoder leads to worse results based on our benchmarks. To alleviate this, we apply *fine-tuning* to the pruned expression encoders to re-calibrate them after parameter removal. This puts task-specific constraints on our pruned model, and improves the performance significantly. Our fine-tuning method is an implementation of *knowledge distillation*, in which the original expression encoder is employed as the teacher. We retrain the pruned encoders so that they can match the teacher's output as closely as possible. Even though the intuition behind the method is simple, our fine-tuned encoders succeed in recovering loss performance after pruning. We anecdotally observe a larger performance gain when fine-tuning with more data, even when the quality of images is not perfect. Fine-tuning positions itself as the logical step after pruning, as it

addresses performance concerns, and allow pruning to solely focus on removing as many parameters from the model as possible.

We combine the results from two previous set of experiments to create the definitive version of NeutrEx-lite. Our final result is composed of a fine-tuned expression encoder with ≈11 million parameters, and a minimal pruned coarse shape encoder with just ≈26k parameters. We conclude our reports, and formally address the research questions raised in Section 1.2.

1, Which of the mentioned research directions: *Pruning*, *Quantization* and *Knowledge distillation* is most suitable to optimize NeutrEx while still maintaining the predictive capability of the quality measure?

Pruning and *Knowledge distillation* are included in our experiments and show promising results in our work.

2, How much can we improve NeutrEx's efficiency with the proposed solutions? What is the acceptable trade off in performance relative to reduced overhead in terms of number of parameters, storage space and inference time?

We are able to reduce NeutrEx's ≈50 million parameters to ≈11 million parameters and still maintain competitive performance. Our experiments show that the performance trade-off is more severe in the expression encoder, while with the coarse shape encoder it is easier to maintain performance to a degree with most of the parameters removed. Our version of NeutrEx-lite is created as a compromise between performance and reduced overhead in mind, as it is not possible to create a version that can maximize both aspects.

3, Would it be possible to combine multiple solutions?

It is possible to combine multiple solutions. Our version of fine-tuning (which is based on knowledge distillation) shows good results when combined with pruning. We recommend to always apply fine-tuning after pruning if the required resources are available.

7.2 FUTURE WORKS

There are multiple potential research directions in our future works, both as an independent research project and in the context of NeutrEx and ISO/IEC 29794-5 [ISO23b].

Out of the three aforementioned methods, we are not able to experiment with *quantization* due to time constraint and prioritization to other methods. Similar to our current setup, we can perform and evaluate quantization independently, as well as in combination with pruning and knowledge distillation. As proven by He et al. [Han+15], a workflow which implements all three methods is possible and can yield good results. Since quantization focuses on improving inference time, we need to extend our benchmarking pipeline to properly accommodate it, e.g including measurements in real time and hardware consumption rate.

As noted in their respective discussions, we are unable to realize some of our ideas for pruning and fine-tuning. For pruning, one option is weighted pruning to remove more parameters from layers that are unimportant, in-

stead of removing from all layers equally. Since NeutrEx only uses the shape code β and pose code θ from the coarse shape encoder's output, it is also an option to prune all parameters related to unneeded outputs such as light code l and camera code c, so that the encoder is fully structured to fit NeutrEx's requirements. For fine-tuning, we can extend the scope of our training, by including more data e.g using the full AffectNet dataset or combine both of our datasets. A more advanced method of fine-tuning which implement knowledge distillation methods from publication in 2.2.3 is also desirable. For example, we can implement a version of deep mutual learning from Zhang et al. [Zha+18], where encoders pruned at different ratios are students to the original encoder.

In the context of utility prediction, we observe a strong disparity in performance between our two benchmarking datasets. Our experiments consistently perform well on Multi-PIE, but less so on the FEAFA+ dataset. This can be attributed to the differences in image quality, as well as variations in capturing scheme. We agree with Grimmer et al. [Gri+23] and raise concern about the lack of datasets for this task. Furthermore, datasets with more diverse demographics such as people of color, women and children are also needed to ensure adequate evaluation. As the purpose of NeutrEx is to eventually be utilized in real life scenarios, more factors that can affect recognition performance such as facial accessories or facial injuries should also be considered, which in turn also require more specialized datasets for further development and testing.

BIBLIOGRAPHY

[AFFOG12] Fernando Alonso-Fernandez, Julian Fierrez, and Javier Ortega-Garcia. "Quality Measures in Biometric Systems." In: *IEEE Security & Privacy* 10.6 (2012), pp. 52–62. DOI: 10.1109/MSP.2011.178.

[BNS19] Ron Banner, Yury Nahshan, and Daniel Soudry. "Post training 4-bit quantization of convolutional networks for rapid-deployment." In: *Advances in Neural Information Processing Systems* 32 (2019).

[Bla+20] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. "What is the state of neural network pruning?" In: *Proceedings of machine learning and systems* 2 (2020), pp. 129–146.

[Blo] Microsoft Open Source Blog. *Olive: A user-friendly toolchain for hardware-aware model optimization*. URL: https://cloudblogs.microsoft.com/opensource/2023/06/26/olive-a-user-friendly-toolchain-for-hardware-aware-model-optimization/.

[Bou+22] Fadi Boutros, Patrick Siebke, Marcel Klemt, Naser Damer, Florian Kirchbuchner, and Arjan Kuijper. "Pocketnet: Extreme lightweight face recognition network using neural architecture search and multistep knowledge distillation." In: *IEEE Access* 10 (2022), pp. 46823–46833.

[CZ18] Shi Chen and Qi Zhao. "Shallowing deep networks: Layer-wise pruning based on feature representations." In: *IEEE transactions on pattern analysis and machine intelligence* 41.12 (2018), pp. 3048–3056.

[CZM18] Ting-Wu Chin, Cha Zhang, and Diana Marculescu. "Layer-compensated pruning for resource-constrained convolutional neural networks." In: *arXiv preprint arXiv:1810.00518* (2018).

[CEKL16] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. "Towards the limit of network quantization." In: *arXiv preprint arXiv:1612.01543* (2016).

[Cho+19] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. "Low-bit quantization of neural networks for efficient inference." In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 3009–3018.

[CBD16] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. *BinaryConnect: Training Deep Neural Networks with binary weights during propagations*. 2016. arXiv: 1511.00363 [cs.LG].

[DEVa] NVIDIA DEVELOPER. *Achieving FP32 Accuracy for INT8 Inference Using Quantization Aware Training with NVIDIA TensorRT*. URL: https://developer.nvidia.com/blog/achieving-fp32-accuracy-for-int8-inference-using-quantization-aware-training-with-tensorrt/.

[DEVb] NVIDIA DEVELOPER. *NVIDIA TensorRT Accelerates Stable Diffusion Nearly 2x Faster with 8-bit Post-Training Quantization*. URL: https://developer.nvidia.com/blog/tensorrt-accelerates-stable-diffusion-nearly-2x-faster-with-8-bit-post-training-quantization/.

[DA18] Orav Anita D'alfonso Alessandro. *Smart Borders: EU Entry/Exit System*. 2018.

[Dam+18] Naser Damer, Yaza Wainakh, Viola Boller, Sven von den Berken, Philipp Terhörst, Andreas Braun, and Arjan Kuijper. "CrazyFaces: Unassisted Circumvention of Watchlist Face Identification." In: *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. 2018, pp. 1–9. DOI: 10.1109/BTAS.2018.8698557.

[DBB22] Radek Daněček, Michael J Black, and Timo Bolkart. "EMOCA: Emotion driven monocular face capture and animation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 20311–20322.

[Das+18] Dipankar Das, Naveen Mellempudi, Dheevatsa Mudigere, Dhiraj Kalamkar, Sasikanth Avancha, Kunal Banerjee, Srinivas Sridharan, Karthik Vaidyanathan, Bharat Kaul, Evangelos Georganas, et al. "Mixed precision training of convolutional neural networks using integer operations." In: *arXiv preprint arXiv:1802.00930* (2018).

[Den+13] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. "Predicting parameters in deep learning." In: *Advances in neural information processing systems* 26 (2013).

[Din+23] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. "Parameter-efficient fine-tuning of large-scale pre-trained language models." In: *Nature Machine Intelligence* 5.3 (2023), pp. 220–235.

[Doc] Pytorch Documentation. *Quantization*. URL: https://pytorch.org/docs/stable/quantization.html/.

[Dod+20] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping." In: *arXiv preprint arXiv:2002.06305* (2020).

[Fab+20] Maël Fabien, Esau Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. "BertAA: BERT fine-tuning for Authorship Attribution." In: *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*. 2020, pp. 127–137.

[Fen+21] Yao Feng, Haiwen Feng, Michael J Black, and Timo Bolkart. "Learning an animatable detailed 3D face model from in-the-wild images." In: *ACM Transactions on Graphics (ToG)* 40.4 (2021), pp. 1–13.

[Gan+22] Wei Gan, Jian Xue, Ke Lu, Yanfu Yan, Pengcheng Gao, and Jiayi Lyu. "FEAFA+: an extended well-annotated dataset for facial expression analysis and 3D facial animation." In: *Fourteenth International Conference on Digital Image Processing (ICDIP 2022)*. Vol. 12342. SPIE. 2022, pp. 307–316.

[Gon+14] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. "Compressing deep convolutional networks using vector quantization." In: *arXiv preprint arXiv:1412.6115* (2014).

[Gou+21] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. "Knowledge distillation: A survey." In: *International Journal of Computer Vision* 129 (2021), pp. 1789–1819.

[Gra+22] Philip-William Grassal, Malte Prinzler, Titus Leistner, Carsten Rother, Matthias Nießner, and Justus Thies. "Neural Head Avatars From Monocular RGB Videos." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 18653–18664.

[Gri+23] Marcel Grimmer, Christian Rathgeb, Raymond Veldhuis, and Christoph Busch. "NeutrEx: A 3D Quality Component Measure on Facial Expression Neutrality." In: *arXiv preprint arXiv:2308.09963* (2023).

[Gro+10] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. "Multi-pie." In: *Image and vision computing* 28.5 (2010), pp. 807–813.

[Guo18] Yunhui Guo. "A survey on methods and theories of quantized neural networks." In: *arXiv preprint arXiv:1808.04752* (2018).

[Guo+19] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. "SpotTune: Transfer Learning Through Adaptive Fine-Tuning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[HMD15] Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." In: *arXiv preprint arXiv:1510.00149* (2015).

[Han+15] Song Han, Jeff Pool, John Tran, and William Dally. "Learning both weights and connections for efficient neural network." In: *Advances in neural information processing systems* 28 (2015).

[HSW93] Babak Hassibi, David G Stork, and Gregory J Wolff. "Optimal brain surgeon and general network pruning." In: *IEEE international conference on neural networks*. IEEE. 1993, pp. 293–299.

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[He+19] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[HX24] Yang He and Lingao Xiao. "Structured Pruning for Deep Convolutional Neural Networks: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), 1–20. ISSN: 1939-3539. DOI: 10.1109/tpami.2023.3334614. URL: http://dx.doi.org/10.1109/TPAMI.2023.3334614.

[He+18] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. "AMC: AutoML for Model Compression and Acceleration on Mobile Devices." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[HZS17] Yihui He, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1389–1397.

[ISO23a] ISO/IEC JTC1 SC37 Biometrics. *ISO/IEC DIS 29794-1 Information Technology - Biometric Sample Quality - Part 1: Framework*. International Organization for Standardization. 2023.

[ISO23b] ISO/IEC JTC1 SC37 Biometrics. *ISO/IEC WD6 29794-5 Information Technology - Biometric Sample Quality - Part 5: Face Image Data*. Intl. Organization for Standardization. 2023.

[Inc22] Apple Inc. *Use Face ID on your iPhone or iPad Pro*. 2022.

[Iso] *Information technology — Biometric sample quality — Part 5: Face image data*. Standard. International Organization for Standardization.

[Int] Neural Network Intelligence. *Pruning Quickstart*. URL: https://nni.readthedocs.io/en/latest/tutorials/pruning_quick_start.html.

[Jia+19] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. "Tinybert: Distilling bert for natural language understanding." In: *arXiv preprint arXiv:1909.10351* (2019).

[Kar+17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive growing of gans for improved quality, stability, and variation." In: *arXiv preprint arXiv:1710.10196* (2017).

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.

[LLC22] Google LLC. *Face Authentication HIDL*. 2022.

[LDS89] Yann LeCun, John Denker, and Sara Solla. "Optimal brain damage." In: *Advances in neural information processing systems* 2 (1989).

[Li+16] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. "Pruning filters for efficient convnets." In: *arXiv preprint arXiv:1608.08710* (2016).

[Li+17a] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. "Learning a model of facial shape and expression from 4D scans." In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36.6 (2017), 194:1–194:17. URL: https://doi.org/10.1145/3130800.3130813.

[Li+17b] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. "Learning a model of facial shape and expression from 4D scans." In: *ACM Trans. Graph.* 36.6 (2017), pp. 194–1.

[Lia+23] Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. "Can Unstructured Pruning Reduce the Depth in Deep Neural Networks?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 1402–1406.

[Lin+20] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. "Channel pruning via automatic structure search." In: *arXiv preprint arXiv:2001.08565* (2020).

[Liu+18] Yongcheng Liu, Lu Sheng, Jing Shao, Junjie Yan, Shiming Xiang, and Chunhong Pan. "Multi-label image classification via knowledge distillation from weakly-supervised detection." In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 700–708.

[Luc+10] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression." In: *2010 ieee computer society conference on computer vision and pattern recognition-workshops*. IEEE. 2010, pp. 94–101.

[MCW15] Debbie S Ma, Joshua Correll, and Bernd Wittenbrink. "The Chicago face database: A free stimulus set of faces and norming data." In: *Behavior research methods* 47 (2015), pp. 1122–1135.

[Mag] Neural Magic. *Part 1: What is Pruning in Machine Learning?* URL: https://neuralmagic.com/blog/pruning-overview/.

[Mah+19] Amirreza Mahbod, Gerald Schaefer, Isabella Ellinger, Rupert Ecker, Alain Pitiot, and Chunliang Wang. "Fusing fine-tuned deep features for skin lesion classification." In: *Computerized Medical Imaging and Graphics* 71 (2019), pp. 19–29.

[MV16] Brais Martinez and Michel F Valstar. "Advances, challenges, and opportunities in automatic facial expression recognition." In: *Advances in face detection and facial image analysis* (2016), pp. 63–100.

[Men+21] Qiang Meng, Shichao Zhao, Zhida Huang, and Feng Zhou. "Magface: A universal representation for face recognition and quality assessment." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 14225–14234.

[Min+18] Chuhan Min, Aosen Wang, Yiran Chen, Wenyao Xu, and Xin Chen. "2pfpce: Two-phase filter pruning based on conditional entropy." In: *arXiv preprint arXiv:1809.02220* (2018).

[Mir+19] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. "Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher." In: *arXiv preprint arXiv:1902.03393* (2019).

[MHM17] Ali Mollahosseini, Behzad Hasani, and Mohammad H Mahoor. "Affectnet: A database for facial expression, valence, and arousal computing in the wild." In: *IEEE Transactions on Affective Computing* 10.1 (2017), pp. 18–31.

[OR+22] D. Osorio-Roig, T. Schlett, C. Rathgeb, J. Tapia, and C. Busch. "Exploring Quality Scores for Workload Reduction in Biometric Identification." In: *2022 International Workshop on Biometrics and Forensics (IWBF)*. 2022, pp. 1–6. DOI: 10.1109/IWBF55382.2022.9794533.

[Pav+19] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. "Expressive Body Capture: 3D Hands, Face, and Body From a Single Image." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[Peñ+21] A. Peña, A. Morales, I. Serna, J. Fierrez, and A. Lapedriza. "Facial Expressions as a Vulnerability in Face Recognition." In: *2021 IEEE International Conference on Image Processing (ICIP)*. 2021, pp. 2988–2992. DOI: 10.1109/ICIP42928.2021.9506444.

[Phi+05] P Jonathon Phillips, Patrick J Flynn, Todd Scruggs, Kevin W Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. "Overview of the face recognition grand challenge." In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 947–954.

[PyT] PyTorch. *Pruning Tutorial*. URL: https://pytorch.org/tutorials/intermediate/pruning_tutorial.html.

[Qia+18] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. "Deep co-training for semi-supervised image recognition." In: *Proceedings of the european conference on computer vision (eccv)*. 2018, pp. 135–152.

[San+19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." In: *arXiv preprint arXiv:1910.01108* (2019).

[SL03] Rebecca Shah and Michael Lewis. "Locating the neutral expression in the facial-emotion space." In: *Visual cognition* 10.5 (2003), pp. 549–566.

[Tan+19] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. "Distilling task-specific knowledge from bert into simple neural networks." In: *arXiv preprint arXiv:1903.12136* (2019).

[Ten] Tensorflow. *Trim insignificant weights*. URL: https://www.tensorflow.org/model_optimization/guide/pruning.

[Too+19] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. "A comparative study of fine-tuning deep learning models for plant disease identification." In: *Computers and Electronics in Agriculture* 161 (2019), pp. 272–279.

[Tou+23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models." In: *arXiv preprint arXiv:2307.09288* (2023).

[Wan+18] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. "Kdgan: Knowledge distillation with generative adversarial networks." In: *Advances in neural information processing systems* 31 (2018).

[Wu+18] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. "Training and inference with integers in deep neural networks." In: *arXiv preprint arXiv:1802.04680* (2018).

[Zha+20] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. "Revisiting few-sample BERT fine-tuning." In: *arXiv preprint arXiv:2006.05987* (2020).

[Zha+18] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. "Deep mutual learning." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4320–4328.

[Zhe+22] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. "I M Avatar: Implicit Morphable Head Avatars From Videos." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 13545–13555.

[Zol+18] Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick Pérez, Marc Stamminger, Matthias Nießner, and Christian Theobalt. "State of the art on monocular 3D face reconstruction, tracking, and applications." In: *Computer graphics forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 523–550.