**Abstract**

Deep neural networks have proved hugely successful, achieving human-like performance on a variety of tasks. However, deep neural networks are computationally expensive, which has motivated the development of *model compression* techniques, which reduce the resource consumption associated with deep learning models.

Nevertheless, recent studies [REF, REF] have suggested that model compression can have an adverse effect on algorithmic fairness, amplifying existing biases in the machine learning models. With this project we aim to extend those studies to the context of affective computing.

To do that, we set up a deep learning classifier to perform facial expression recognition and implement several model compression techniques on top of it. We then run experiments to examine the individual and combined effect that compression techniques have on the model size, accuracy and fairness. Our experiments confirm results from previous studies that certain compression techniques can amplify algorithmic biases. However, we also find that the effect on fairness varies widely across different compression strategies.

**1. Introduction**

**1.1. Motivation**

Recent years have seen deep neural networks (DNNs) achieve state-of-the-art performance on a variety of problems including face recognition [[REF](https://ora.ox.ac.uk/objects/uuid:a5f2e93f-2768-45bb-8508-74747f85cad1)], cancer detection [[REF](https://www.nature.com/articles/s41568-020-00327-9?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+nrc%2Frss%2Fcurrent+%28Nature+Reviews+Cancer+-+Issue%29)], natural language processing [[REF](https://ieeexplore.ieee.org/abstract/document/8416973)], etc. Deep learning has proved particularly effective at extracting meaningful representations from raw data [[REF](https://academic.oup.com/bib/article/22/2/1902/5826499?login=true)].

However, as the predictive performance of deep neural networks has increased, so has the size of deep learning architectures: Modern DNNs can consist of hundreds of millions of parameters [[REF](https://www.mdpi.com/1999-5903/12/7/113/htm)], making them slow to train and hard to store. Deep learning’s growing computational cost has made it hard to deploy deep learning models on resource-constrained devices (e.g. mobile phones, robots, microcontrollers) which often lack the storage, memory or processing power to large DNNs [REF, REF]. The high resource consumption associated with deep learning models has also been problematic in the light of initiatives such as the “Green-AI” [[REF](https://spectrum.ieee.org/energywise/artificial-intelligence/machine-learning/energy-efficient-green-ai-strategies)] movement advocating for a reduction in the carbon footprint and the environmental impact associated with artificial intelligence.

This has given rise to the development of *model compression* strategies, aiming to reduce the computational cost associated with deep learning models. Examples of model compression techniques include model pruning [REF], quantization [REF], weight clustering [REF], etc. We provide a more detailed overview of the compression strategies considered in this project in Chapter [CH].

However, a couple of recent studies have suggested that model compression can amplify biases in machine learning: Hooker et al. [[REF](https://arxiv.org/abs/2010.03058)] demonstrate that pruning and post-training quantisation can amplify biases when classifying hair colour on CelebA. This issue is also raised in a study by Paganini [[REF](https://arxiv.org/abs/2009.09936)] who discusses the effect of pruning on algorithmic fairness and proposes a framework for fair model pruning.

**1.2. Project Goals**

With this work, we aim to extend the aforementioned studies by Paganini and Hooker et al. to the context of affective computing. In particular, we consider the task of *facial expression recognition* (FER) where the model has to classify expressions based on images of human faces.

To this end, we train an FER model on the CK+ dataset and implement three compression strategies (pruning, weight clustering and post-training quantisation) on top of it. We then evaluate and compare the performance of the baseline model vs the performance of the compressed models, and analyse the results to address three research questions:

* **RQ1: “How effective are different compression techniques in the context of FER?”** I.e., can model compression achieve a considerable reduction in the model size, while preserving a high level of predictive accuracy?
* **RQ2: “Do model compression techniques amplify biases?”** Here we seek to verify the claims by Paganini and Hooker et al. across a wider variety of compression techniques and in the context of FER.
* **RQ3: “Is the impact on fairness identical across different compression techniques?”** We are interested to know whether all compression strategies amplify biases to the same extent.

**1.3. Contributions**

This study extends the previous work by Paganini and Hooker et al. in three directions:

* **Extending the problem to affective computing:** We consider the problem of compression’s effect on fairness in the context of affective computing and, in particular, facial expression recognition on CK+. By comparison, the study by Hooker et al. is based on classifying hair colour on MNIST, and the study by Paganini et al. considers object recognition and digit classification tasks (on CIFAR10, CIFAR100, MNIST, SVHN and other datasets).
* **Considering more model compression techniques:** Our study involves three compression strategies (pruning, weight clustering and post-training quantisation) – one more by Hooker et al. (who consider pruning and post-training quantisation), and two more by the study by Paganini, which focuses solely on pruning.
* **Considering the combined effect of compression techniques:** In practice, compression techniques are often combined together to form so called “compression pipelines” [REF]. That is why, we also consider combinations of compression strategies (pruning with quantisation and weight clustering with quantisation). The previous studies have only examined the individual behaviour of compression techniques.

**2. Background**

**2.1. Fairness**

Nowadays, machine learning algorithms are used to inform or automate decision-making across various fields of high social importance. Machine learning approaches have been used for automating recruitment in large companies [REF], assigning credit scores [[REF](https://www.sciencedirect.com/science/article/pii/S0040162520311355)] and anticipating criminal activity [[REF](https://www.technologyreview.com/2020/07/17/1005396/predictive-policing-algorithms-racist-dismantled-machine-learning-bias-criminal-justice/)] to name a few.

The increasing impact of machine learning on our society has highlighted the importance of *algorithmic fairness*. An algorithm is considered to be fair if its behaviour is not improperly influenced by *sensitive attributes* (e.g., a person’s gender or race) [REF].

Nevertheless, recent studies have exposed the propensity of algorithms to be unfair and exhibit dangerous biases, potentially “reinforcing the discriminatory practices in society” [[REF](https://ojs.aaai.org/index.php/ICWSM/article/view/3255)]. For example, Amazon’s AI recruitment tool has been reported to favour male applicants over female applicants [[REF](https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G)]. Apple’s credit score has also been shown to systematically disadvantage women [[REF](https://www.bbc.co.uk/news/business-50365609)]. A study by Joy B. [GENDERSHADES] has demonstrated that popular facial analysis services perform disproportionately poorly on darker-skinned females.

The increasing awareness of algorithmic biases has given rise to multiple fairness initiatives such the Algorithmic Justice League [REF], Google’s ML Fairness [REF] and IBM’s AI Fairness 360 program. Despite the large body of research which has studied the problem, though, there is still no consensus in the scientific community on what the precise definition of fairness should be. Multiple definitions of fairness have been proposed but none of them is a “silver bullet” that fits all use cases. Instead, the “right” choice of a fairness metric often depends on the specific context in which the algorithm is used.

For this project, and in the context of facial expression recognition, we adopt the fairness definition of *overall accuracy equality*. The definition is akin to the ideas of predictive parity [REF] and disparate mistreatment [[REF](https://arxiv.org/pdf/1610.08452.pdf)], and states that a fair algorithm should have the same predictive accuracy regardless of any underlying sensitive attributes.

To express this formally, assume we have a facial expression recognition model that aims to predict a subject’s *true* expression Y by producing a *prediction* Yhat. Let the subject’s gender be denoted by G and be equal to either m (male) or f (female) [CLARIFY]. In that case, we expect that a fair model would have the following property:

That is, we expect a fair FER model would classify the expressions of male and female subjects with the same accuracy.

For this project and in the context of facial expression recognition, we adopt the formulation of overall accuracy equality. The definition is akin to the ideas of predictive parity and “disparate mistreatment” (can be found at <https://arxiv.org/pdf/1610.08452.pdf>), and states that a fair algorithm should have the same predictive accuracy regardless of any underlying sensitive attributes.

To express this formally, assume we have a facial expression recognition model that tries to predict the true emotion of a given subject (Y) and outputs a prediction Y^{hat}. Let the subject’s gender be denoted by G and be equal to either m (male) or f (female) [CLARIFY]. Then, we expect that a fair model would have the following property:

P(Y = Yhat bar G = m) = P(Y = Yhat bar G = f)

That is, we expect a fair FER model would classify the expressions of male and female subjects with the same accuracy.

**2.2. Model Compression Techniques**

Below, we provide a summary of the compression techniques we use throughout the study.

**2.2.1. Quantisation**

Quantisation is a popular compression method which can significantly reduce the size of the model, leading to savings in storage and memory [REF]. The key idea behind quantisation is sacrificing precision for efficiency – while most standard DNN implementations represent weights and activations using the float32 datatype, quantisation allows representing those values using a smaller datatype – normally float16 or int8 [REF].

Quantisation can either be introduced *during* training (also known as “quantisation-aware training”), or it can be applied to a trained model, which is known as post-training quantisation.

**2.2.2. Pruning**

Weight pruning is another compression strategy which can greatly reduce the size of a DNN [REF]. It does so by eliminating redundant weights which contribute little to the behaviour of the model. As illustrated in Figure [FIG], pruning reduces the density of the neural network, making it more lightweight and easier to compress by traditional compression tools such as zip [[REF](https://www.tensorflow.org/modeloptimization/guide/pruning)].

Which weights get pruned is dictated by the pruning strategy. The most popular approach, which we focus on in this project, is called *magnitude-based pruning* [REF] and eliminates the weights with the lowest absolute value – i.e., the ones whose values are the closest to zero. The proportion of weights that need to be pruned is called the pruning *sparsity* – e.g., pruning at 90% sparsity would remove 90% of the connections in a given network.

2.2.3. Weight Clustering

Weight clustering (also known as weight sharing [REF]) reduces the size of the model by grouping together weight of similar values. This process is illustrated in Figure [FIG]: During step (1), weight matrices are processed by a clustering algorithm, which maps each weight to one of n clusters (where n is the number of clusters specified by the user). Each cluster consists of an index ({0, 1, ..., n-1}) and a *centroid* value which is representative of the values of the weights in the cluster.

During step (2), the weight matrices of the DNN get replaced by pull indices – instead of containing the values of the weights, pull indices contain the index of the cluster corresponding to each weight. During inference, the DNN model can use the pull indices to dereference obtain the centroid values representing each weight.

The new model is more efficient for two reasons: First, float values only need to be stored to represent the centroid values of each of the n clusters, while the weight matrix is replaced by pull indices, each index represented by the smaller integer type. And second, the resulting pull indices are more likely to contain repeating values, making standard compression tools (e.g., zip) more effective, similar to pruning.

3. Implementation

3.1. Data

To perform facial expression recognition, we need a dataset of human faces. To this end, we use the Extended Cohn-Kanade Dataset (CK+) dataset [REF] which has been widely used in the context of facial expression recognition [REF, REF]. CK+ contains 327 labelled image sequences across 123 unique subjects, expressing one of 8 emotions - neutral, anger, contempt, disgust, fear, happy, sadness, surprise.

Furthermore, we need annotations of demographic attributes to examine bias. CK+ does not provide any annotations in that respect, so we manually annotate all 123 subjects based on their gender appearance. We assign a value “male” if the subject looks masculine, and a value “female” if they look feminine. The annotations are provided in the project repository under [FILENAME]. According to our annotation, the dataset consists of 84 female subjects and 39 male subjects.

We then apply several pre-processing steps to the CK+ dataset, which are documented in the CKPlusPreprocessing.ipynb Jupyter notebook **(NOT THE ACTUAL SCRIPT)**: For each sequence in the dataset, we take the first frame to represent a neutral emotion, and the last 3 frames to represent the emotion which the sequence was annotated with (e.g. “happy”, “sad”, etc.). That is a common pre-processing step since CK+ sequences “are from the neutral face to the peak expression” according to the dataset’s documentation.

We then use the dlib libraries to detect the faces of the subjects and crop the images around them (allowing an extra 10% on each side to avoid cropping out parts of the chin, forehead or ears). This pre-processing step aims to facilitate our FER classifier by removing potential distractions from the background.

For validation purposes, we split the original CK+ dataset into a train and test dataset. We use cross-subject validation, allocating 86 subjects to the train dataset and the other 37 to the test dataset. That gives us 924 images in total in the train dataset and 384 images in the test dataset. The train and test dataset follow a similar distribution with respect to the emotion labels as shown in Figure [FIG].

Finally, before being fed to the classifier, several data transformations are applied to the images using Keras’ ImageDataGenerator [REF]: Images are scaled down to 48x48 pixels and converted to grayscale (since CK+ contains some RGB images). Finally, to compensate for the relatively small size of the dataset, we augment the data by applying random horizontal flipping and random rotation in the range [-10 degrees, 10 degrees].

3.2. Baseline Model

We implement an FER classifier as a baseline to which we will apply the compression strategies. We follow a tutorial on facial expression recognition by S. Kekre [[REF](https://www.coursera.org/projects/facial-expression-recognition-keras)] to set up a DNN classifier in Keras [REF]. The exact architecture is shown in Table [REF] and is inspired by a study by Goodfellow et al [[REF](https://arxiv.org/abs/1307.0414)]. It contains 4 convolutional layers, followed by 2 hidden fully-connected layers (with pooling and dropout layers inbetween). At the time of publication, architecture achieved a then state-of-the-art performance of around 65% on the FER-2013 dataset [[REF](https://www.researchgate.net/publication/330611486_Using_CNN_for_facial_expression_recognition_a_study_of_the_effects_of_kernel_size_and_number_of_filters_on_accuracy), [REF](https://arxiv.org/abs/1307.0414)].

We then compile the baseline using an Adam optimiser [[REF](https://arxiv.org/abs/1412.6980)] and a categorical cross-entropy loss [[REF](https://arxiv.org/abs/2011.05231)]. We train the neural network for 20 iterations keeping track of training and validation accuracy, and training and validation loss (where validation is performed on the test dataset and training is performed only on the test dataset). At every iteration, we store the “best” weights so far (i.e. the ones associated with the highest validation accuracy) as an \texttt{.h5}.

The training process is visualised in figure [FIG]. We can see that both training and validation accuracy are increasing (and training and validation loss decreasing) up until the 10th iteration, after which the training accuracy keeps increasing but the validation accuracy gradually drops. This is a sign of overfitting the model, so throughout our experiments we use the weights obtained after the 10th iteration when the model achieves peak validation accuracy of 67.96%.

3.3. Model Compression

We implement three model compression strategies – *magnitude-based weight pruning*, *post-training quantisation* and *weight clustering*. To do this, we make use of TensorFlow’s Model Optimization Toolkit [REF], part of the TFLite framework [REF].

To quantise the model, we convert the pre-trained Keras baseline described in the last section to a TFLite model and apply the default TFLite optimisation strategy [[REF](https://www.tensorflow.org/api_docs/python/tf/lite/Optimize)] which reduces the model representation to 8 bits. Finally, we store the quantised model on disk and compress it using the \texttt{zip} compression tool, so that we are able to observe the change in size that quantisation has introduced.

We apply pruning using TFLite’s ConstantSparsity schedule [[REF](https://www.tensorflow.org/model_optimization/api_docs/python/tfmot/sparsity/keras/ConstantSparsity)]. Our implementation is parameterised by the pruning sparsity – we observe the effect of this parameter on compression in Section [SECTION]. After pruning has been applied, we fine-tuned the pruned model for 2 iterations as suggested by the TFLite documentation. Similarly to quantisation, we store the model on disk and compress it to evaluate the reduction in size.

We implement weight clustering via TFLite’s \verb|cluster\_weights| module and parameterise it by the number of clusters. Similar to pruning, we fine-tune the clustered model, store it on disk and compress it.

4. Experiments

4.1. Metrics

In our experiments, we compare the uncompressed baseline model against its compressed versions using 3 main metrics:

* **Model size** – that is the size of the model on disk in megabytes. This metrics measures how effective a compression strategy is in reducing the storage requirement of the model. While some of the compression strategies could also reduce other systems metrics such as latency, we use model size since all three compression techniques are primarily used to reduce storage consumption.
* **Accuracy** – this is the overall accuracy of a model (i.e., number of correctly classified images over the overall number of images). We use this metric as an indicator of how compression has impacted predictive accuracy.
* **Female accuracy and male accuracy** – to examine the fairness of the models, we also introduce the measures of female and male accuracy. We define female accuracy as the number of correctly classified images containing a female subject over the total number of images containing a female subject. Similarly, male accuracy is the number of correctly classified images where the subject was male over the total number of images where the subject was male. Under the overall accuracy equality formulation of fairness, which we defined in section [SECTION], an unbiased model should have equal or similar female and male accuracy metrics. Conversely, a large discrepancy between the model’s accuracy for males and females would be a strong indicator of algorithmic bias

4.2. Baseline Performance

We first report the performance of the baseline model to establish a basis of comparison against the compressed models. As mentioned previously, the baseline model reports an overall accuracy of **67.96%** on the test dataset. We measure the baseline’s size in the same way we measure the size of the compressed models – we save the model as a file on disk and apply \texttt{zip} compression to the file. After doing that, we find that the baseline’s model size on disk is 16,512,048 bytes or around **16.51 megabytes**.

We find that the female accuracy of the baseline model is **67.08%** and the male accuracy is **69.44%.** This is interesting because as we mentioned in Section [REF], the CK+ dataset is imbalanced in favour of female subjects and therefore we would expect the baseline model to classify females more accurately.

One reason why the classifier might perform slightly better on male faces is the slight difference in the distribution of emotions across males and females in CK+ – for example, only 2.1% of male subjects have expressed “contempt” while female subjects have expressed this emotion more than twice more frequently (5.02%). If contempt is an emotion that is inherently harder to classify, then this difference could translate into a minor advantage for classifying male subjects. In any case, though, the gap between male and female accuracy is too minor to conclude the baseline model is biased.

4.3. Quantisation Results

We evaluate quantisation by quantising the baseline model three times and reporting the mean values for each metric. After applying quantisation to the baseline model, we observe a 4x reduction in the model size as illustrated in Figure [FIG].

Moreover, this compression comes at no cost – there is no change in the predictive accuracy or fairness whatsoever: Overall accuracy, female accuracy and male accuracy have remained completely identical with those of the baseline model. Quantisation preserves the fairness and predictive accuracy of the model, while introducing a significant reduction in model size.

4.2. Pruning Results

Tried with 10, 20, 30, 40, 50, 60% sparsity. Repeated each experiment 3 times.

trade-off – bigger sparsity 🡺 smaller model.

However, model suffers from big sparsity and if sparsity too big the model is of no use.

IN GENERAL

size reduction – good. (How much – in times)

accuracy – doesn’t seem to be severely impacted.

but then look at the fairness metrics – woah! The gap has grown massively.

As mentioned previously, we also implemented two combined compressions – one of which pruning + quantisation.

Results show that quantisation can decrease the size of the pruned model a further 3.5 times on average.

Meanwhile no impact on predictive performance or fairness – the differences in overall accuracy, female accuracy and male accuracy are all well under 1% compared to the values of the pruned model.

* Quantisation comes for free again.

4.3. Weight clustering

size reduction – look at the graph again.

**MENTION THE EFFECT OF COMBINING PRUNING!**

WHAT CAUSES BIASES?

* class imbalance
* class complexity

WHAT CAUSES PRUNING (AND MODEL COMPRESSION IN GENERAL) TO AMPLIFY BIASES?

* DNNs are not well understood (theoretically) because they are essentially black boxes.
* But it has to do with network capacity being reduced (that’s mostly relevant for pruning though – not as much for, say, quantisation).

We are interested in quantifying the inequality of treatment among classes, cohorts, and individuals as **network capacity is reduced**.

Specifically, the hypothesis we seek to test is that **class imbalance** and **class complexity** affect the per-class performance of pruned models.

***The minimal changes to overall accuracy hide disproportionately high errors on a small subset of examples.***