Emotion Engine – Micro-Expression Recognition with Deep Learning

Leo Davidov
leo.davidov@student.oulu.fi

Raffaele Sali
raffaele.sali@student.oulu.fi

Sajjad Ghaeminejad
sajjad.ghaeminejad@student.oulu.fi

Zhou Yang
zhou.yang@oulu.fi

Anatolii Fedorov
anatolii.fedorov@student.oulu.fi

Timofei Polishchuk
timofei.polishcuk@student.oulu.fi

TA in charge: Yante Li
The Center for Machine Vision and Signal Analysis
University of Oulu, Finland
yante.li@oulu.fi

Abstract

Facial micro-expressions are spontaneous, subtle, and rapid muscle movements, which can be used to discover the true feelings that humans may hide. The primary goal of the project is to investigate how different Deep Learning AI model architectures perform in micro-expression recognition considering different feature representations derived from the facial image sequences, and in parallel train the best performer and prepare the video processing and annotation pipeline. Project also considers recognition and classification of facial micro-expression action units (AUs) from the feature sequences as an auxiliary target for our models.

Developing reliable AU and micro expression recognition models is very challenging because the signals are brief, low-amplitude, noisy, and amount of training data is very limited. We tried to tackle this problem by utilizing two high quality ME research datasets CASME II [1] and 4DME[2] and applying state-of-art image and signal analysis methods incorporated into MEB [3, 4] — Eulerian Motion Magnification, Gunnar Farnebäck's optical flow, with consequent computation of optical strains using Sobel kernels, as well as Local Binary Patterns (LBPs), and block-wise 3D Fast Fourier Transform. Mainly three multitask deep architectures were trained and compared. 3DCNN operating singly on optical flow computed from temporally normalized facial image sequence representation appeared as the strongest performer, reaching up to 61% in emotion-class accuracy on a subject-disjoint evaluation set, which is competitive with reported state-of-the-art accuracy results for the corpora of such size. The resulting models power an end-to-end pipeline that can annotate raw videos after RetinaFace-based face detection and alignment, demonstrating practical viability despite limited data. Code is publicly available in https://github.com/littlemicrowave/ME-based-emotion-classifier.

Keywords: micro-expression recognition, Eulerian motion magnification, optical flow, multitask learning, action units

1 Introduction

Micro-expressions (MEs) are brief (often <200 ms), low-amplitude facial movements that can reveal true emotions, feelings, and even thoughts, despite attempts to mask them, since they are caused unintentionally by corresponding brain and nervous system activity impulses. Since these movements involve subtle, short-lived muscle activations, they are difficult to capture and annotate reliably. The Facial Action Coding System (FACS) can additionally provide an action-unit (AU) vocabulary that is commonly used to describe such events without commitment to emotion labels [5]. In this research field there are several spontaneous ME datasets available, most notable are — SMIC, CASME II, and SAMM, which have catalyzed progress but remain limited in size, subject diversity, and cross-dataset consistency, which complicates supervised learning and evaluation [1].

In this project, our goal was to systematically compare deep learning architectures for sequence processing across multiple spatiotemporal feature families. We leveraged Eulerian Video Magnification (EVM) to amplify subtle facial dynamics while preserving spatial layout [6], to capture dynamic cues, we employed a Gunnar Farnebäck formulation when computing flow fields [7], and 3D Fast Fourier Transform (FFT), which not only analyzes rapid changes of ME in the frequency domain, but also leverages the spatial and continuous temporal information [8].

Concretely, we gained access and used two datasets— CASME II [1] and smaller version of 4DME [2], created five sets of features per short face clip and combined same feature representations from both datasets into joint feature sets.

In our experiments was tried:

1. First applying EVM on the raw frames, then temporally normalizing the resulting clips, and after extracting the Farnebäck's optical flow and optical strains.
2. EVM of raw frames, and consequent temporal normalization.
3. Temporal normalization and consequent dense optical flow extraction using Farnebäck's method and optical strain computation.
4. Early fusion of previously computed optical flow and optical strains with frame-wise LBPs computed from the temporally normalized motion magnified clips in case 2).
5. Spatiotemporal frequency-amplitude spectrum computed using 3D Fast Fourier Transform on the temporally normalized motion magnified clips in case 2).

Difference between set 1) and set 3) is inclusion of motion magnification before optical flow vectors extraction.

We then designed several model architectures, mainly two model families: a 3D-CNNs and a 2D-CNN image encoder with bidirectional LSTM sequence processing model.

2 Related work

MER pipelines typically combine careful face normalization with motion-sensitive representations and subject-independent evaluation. Studies emphasize how dataset biases and protocol choices, for instance, leave-one-subject-out and cross-dataset markedly affect reported performance, motivating designs that are robust to class imbalance and domain shift. Research has coalesced around a few core resources—SMIC, CASME/CASME II, and SAMM—whose elicitation procedures and annotation taxonomies differ and thus induce distribution shift; newer corpora such as 4DME further stress generalization by offering additional subjects and recording conditions [9].

To increase the signal-to-noise ratio before feature learning, many systems amplify the subtlest facial motions with Eulerian Video Magnification and then estimate dense motion fields, TV-L1 optical flow, in particular, is a common choice in prior work because of its robustness to outliers and piecewise-constant motion [10], our pipeline instead adopts the Farnebäck's formulation. Early ME recognition relied on hand-crafted spatiotemporal descriptors such as LBP-TOP and its variants, which remain competitive on small datasets but are capacity-limited compared to modern deep video models [11]. With larger-scale pretraining and 3D convolutional architectures (e.g., C3D, I3D, and 3D-ResNets), end-to-end spatiotemporal encoders have become a backbone for short facial clips, often outperforming shallow descriptors while preserving temporal locality. ME-specific deep approaches frequently inject motion explicitly—using optical-flow streams or apex-centric cues (e.g., OFF-ApexNet)—to steer the network toward genuine facial micromovements rather than contextual artifacts [12]. Across these pipelines, accurate face localization and alignment are crucial; single-stage detectors such as RetinaFace are widely adopted to stabilize geometry, standardize, crop and scale facial region before temporal modeling.

Overall, prior works suggest that (I) magnification and optical flow can enhance ME salience, (II) 3D CNNs are strong spatiotemporal encoders when data are scarce but structured, and (III) evaluation under subject- or dataset-disjoint splits is essential to gauge real-world generalization, yet few systems unify these ingredients while jointly predicting both emotions and AUs across multiple datasets. Those gaps directly shaped the design and validation strategy of our multitask approach.

3 Methods

3.1 Datasets and preprocessing

Our version of CASME II consisted of 256 clips recorded at 200 FPS, with events as short as 0.02 s and at most 0.75 s, with seven emotion labels – happiness, others, disgust, repression, surprise, fear and sadness, and 19 action unit categories. The best reported five-class accuracy especially only on CASME II hovers around 63.41% [1]. 4DME adds 267 clips at 60 FPS, with duration from 0.13 s up to 2.3 s each, with five primary micro-expression categories same as in CASME II – happiness, others, disgust, repression, and surprise, along with doubles such as "surprise+negative" and 20 annotated action unit categories. Our versions of CASME II and 4DME datasets have already contained standardized, and aligned facial images, ready to work with. CASME II clips were in 3 channels RGB, while 4DME consisted of single channel, grayscale images.

MEB was used as a base for datasets manipulation and feature extraction, for instance, MEB was mainly targeted for loading the datasets, loading annotations, subject ids, resizing, and gray scaling the images in CASME II. MEB framework has also provided methods for temporal normalization. We have actively employed uniform temporal sampling as the method for temporal normalization to standardize the duration of clips to the uniform frame count, since duration of clips widely varied between episodes as well as between datasets. Uniform temporal sampling is the simplest way to align clips which are recorded in completely different frame rates and have completely different duration, deep learning models on the other hand learn the general spatiotemporal dynamics which are represented in uniform sequences. This method has some disadvantages – the loss of unique temporal dependency within episode, in other words we are fitting random time intervals into predefined number of frames.

MEB framework also provided a reference for the py_evm, which is an implementation variant of Eulerian Motion Magnification. Parameters which are used in py_evm Eulerian Motion Magnification method – alpha 10, R1 0.47, R2 0.1, number of levels 6, lambda 16. We set R1 to 0.47 according to the Nyquist frequency (max 0.5), if, target FPS would be 30, then frequencies up to 14 Hz are considered, this is quite tricky, since higher the upper bound frequency the more noise is getting amplified:

1. R1 – upper bound frequency cutoff, normalized
2. R2 – lower bound frequency cutoff, normalizied, thus, we amplify everything in between.
3. Alpha is amplification coefficient
4. Number of levels – number of Laplacian Pyramid levels, each level is the difference between two consecutive Gaussian Pyramids, after lower in upsampled.
5. Lambda controls how alpha will be applied to each level, higher the lambda for more it will amplify higher levels of Laplacian pyramids, thus focus on more subtle actions.

For optical flow computation, we modified the approach presented in MEB to improve computational efficiency. Specifically, instead of using the original "classic+nl-fast" optical flow variant implemented through the MATLAB engine which caused significant performance overhead, we adopted OpenCV's implementation of Gunnar Farnebäck's optical flow. This modification allowed for a more efficient, fully Python-based pipeline. Explanation of parameters in OpenCV Farneback method (0.5, 4, 15, 4, 5, 1.2), considering the order:

1. Pyramid Scale – image scale (<1) between pyramid levels (usually 0.3–0.8), smaller scale (0.3), more global motion, smoother flow, the larger, more detail, possibly more noise.
2. Levels – number of pyramid levels, more levels (5–6) the better handling of large motion, smoother overall flow.
3. Window size – averaging window size, the larger window, more smoothing, less noise, but less detail and slower response to motion.
4. Iterations – iterations at each pyramid level, the more iterations, more stable convergence, slightly smoother results (usually 3–5).
5. Poly_n – size of the pixel neighborhood used for polynomial expansion (usually 5 or 7), the larger, smoother, denoised motion field, if smaller, more detailed but noisier.
6. Poly_sigma – standard deviation of Gaussian used for polynomial smoothing, higher (1.5), more smoothing, denoising, the lower (1.1), more detail, more noise.

The optical strain computation followed the method from MEB, which estimates local deformation in the optical flow field using Sobel kernels. The strain magnitude is derived from the spatial gradients of the horizontal "u" and vertical "v" flow components.

The block-wise 3D Fast Fourier Transform computation was implemented using functions provided by the NumPy library, with an additional optimized variant developed using TensorFlow tensors for improved computational efficiency. The 3D FFT is computed on spatiotemporal blocks of size 8x8x8 in a straightforward manner. First, a 1D FFT is performed along the temporal axis for each pixel's intensity sequence, resulting in a matrix of shape (F, H, W), where each element represents the two-sided amplitude spectrum of temporal intensity oscillations at that pixel. Next, to analyze how these frequency amplitudes vary across the horizontal dimension, an additional FFT is computed along the width axis while keeping the vertical position fixed. This step produces a 2D FFT, capturing frequency decomposition of pixel intensities across time and width. Finally, to examine how these row-wise frequency amplitudes vary across the vertical dimension, another FFT is applied along the height axis, yielding the final 3D amplitude spectrum of size (8x8x8). This tensor represents spatial–temporal oscillations of pixel intensities across both space and time within the block. The computation is performed block-wise to preserve local motion characteristics from distinct facial regions. Each image (224x224) is divided into 28x28 non-overlapping blocks, and the 3D FFT is applied to each block independently. The resulting block-specific spectra are then concatenated to form a composite feature representation, providing fine-grained, region-specific spatiotemporal representation of subtle facial motion. Example is presented in a figure below.

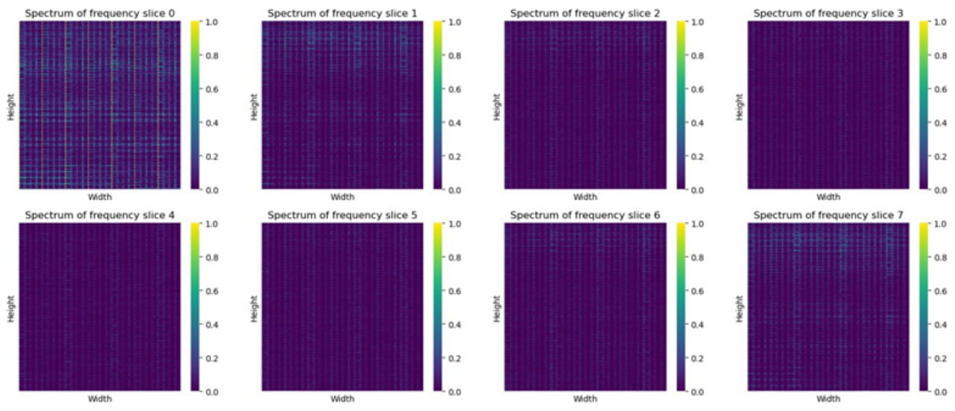

Figure 1: 3D FFT decomposition of clip

For the sort of fair model tuning and evaluation was developed a separate sampler function which provides following functionality – it can build classical random train test splitting of the data in a stratified manner, or it can sample specified number of subjects from the dataset for testing, as well in stratified manner, based on L1 distance, in other words script tries to pick N subjects from the dataset so that emotion class balance in the testing set remains as close as possible to the original dataset. Additionally, sampler can consider percent of data which testing subjects should cover, both datasets in total had only 63 subjects. Script was allowed to have a tolerance of picking +/-5%, since sometimes it is impossible to build a good split considering stratification, N subjects, and percent size of testing sample.

The deployment script then mirrors some of steps presented above. RetinaFace finds and aligns faces in raw videos, then those are resized to 224x224, and converted to grayscale, sliced into eight-frame windows, and Farnebäck's optical flow is then computed, so the trained network sees the same representation it was optimized on.

3.2 Model architectures

To reach the state of models as they are a numerous amount of architectural tweaks, kernel size changes, stride changes, a variety of regularization techniques, class weightings, consequent retraining, evaluations and updates was applied, for instance, 3DCNN model was updated at least 500 times in order to get the desired result, at some point the focus was mainly put on 3DCNN architecture since it initially showed more promising results, so it provided a good starting point for the further updates.

All models consider two heads, AU head and emotion head. For AU head was used binary focal crossentropy loss to address sparsity and imbalance of binary multilabel AU target vectors. For emotion classification we used weighted sparse categorical cross-entropy loss. Loss weights were computed based on the inverse frequency of class labels. For optimization was used Adam with a 1×10^{-4} learning rate. Losses from both heads are combined with following weights 1 for emotions and 2 for action units.

Our 2DCNN + bidirectional LSTM is a hybrid model of 1.7 million parameters which under "evolution" received a block of multiple 3D convolutional layers for low-level spatiotemporal feature extraction, idea of 3D block is to capture short motion cues and spatial patterns before Time Distributed 2DCNN frame encoder. 2DCNN encoder which is implemented is ResNet style with residual connections for the proper gradient flows is used then for per-frame spatial feature extraction, representation and compression, with final layer of global max pooling which reduces each frame's representation to its most salient features with a consecutive Bidirectional LSTM for temporal dependency modeling across frames. Concise representation of the model is presented on the image below.

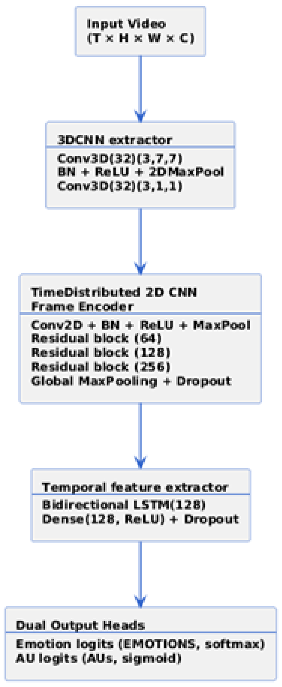

Figure 2: 3DCONV + TD2DCNN encoder + biLSTM

It was noticed that a single time distributed 2D frame encoder block alone with LSTM cannot achieve any proper results, presumably since it misses those motion dynamics which can be extracted by 3D convolution across time and space.

The 3DCNN consisted of 1.2 million parameters and inherited the ResNet style with residual connections, but the better performance was achieved when residual blocks incorporated layer skipping connections with full 3x3x3 kernels, it presumably smoothens out the noise on the main path and acts as additional downstream feature extractor. The backbone of our 3DCNN progressively compresses clips into a compact single vector representation. The network begins with two 3D convolutional layers (8 and 16 filters) that capture low-level motion and appearance cues, followed by a series of residual modules with kernels 3x3x3 and increasing filter sizes (32, 64, 128, 256) and strides of 2 for deeper spatiotemporal feature extraction, ending up with flatten layer which outputs 1024-dimensional vector which encodes a sequence of 8x224x224. Dropout and spatial dropout layers are added for regularization and to reduce overfitting. Incorporation of spatial dropouts in the initial phase of compression significantly increased the ability of model to generalize the emotions of unseen subjects, the difference between dropout and spatial dropout is that it completely drops entire feature channels within the convolutional tensors, instead of single values like it does the normal dropout. Compact model view presented below:

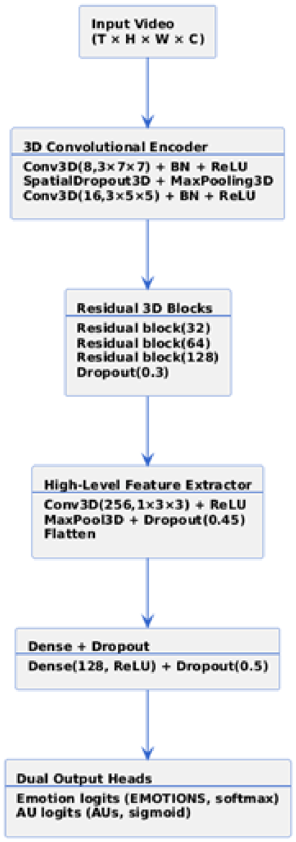

Figure 3: 3D CNN

The 3DCNN model for 3DFFT had 1.3 million parameters and followed pretty the same structure as above but it incorporated more aggressive spatial compression and was begging with wide 3x8x8 kernel and stride 2x8x8 16 filter convolutions, since our 3DFFT was computed for blocks of size 8x8x8. After max pooling downsampling residual 3D blocks follow, with increasing filter sizes 64, 128, 256 and small temporal kernels, which enables moderate spatiotemporal compression speed and feature extraction, the output embedding of the 3DFFT spectrum is the same 1024-dimensional vector as in the model above. Below is the brief summary of 3DFFT encoder:

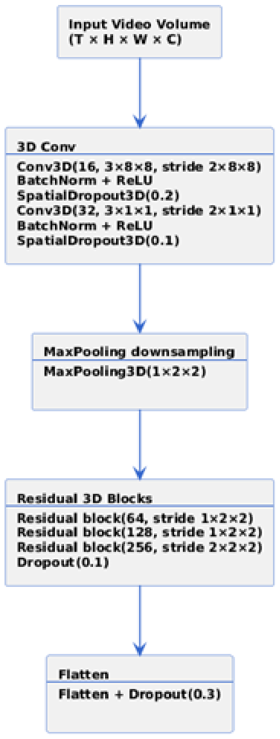

Figure 4: 3DCNN for 3DFFT

In all of our models we mostly used strides for the downsampling instead of max or average pooling, this enables downsampling step to act as a separate feature extractor, we also incorporated batch normalizations mostly after each convolutional layer. During training, many additional strategies were utilized to avoid overfitting such as learning rate reduction on plateau, learning rate schedulers, and exponential learning rate reduction. It was noticed that when the model size is around 1.2 million it balances the ability to learn and avoid rapid overfitting we also tried both batched and non-batched inputs, with batched inputs learning was slightly more stable. Also were tested kernel regularizations L1 and L2, first tries to minimize sum of squared weights, and second one sum of modulus of weights, no improvements so far were noticed, models just stops learning completely despite minimal coefficients were set. We have also tried to use 112x112 images, it was assumed that it could reduce noise, but instead of improvement it did just the opposite, so 224x224 appeared to optimal image size. To sum up, we tried to do models as small as possible, and regularize them as much as possible.

4 Experiments

As it was stated we have created five feature sets, first we evaluated the models performance on the classical random train test split, and then further picked the best performer for evaluation and tuning on the so called subject-disjoint “Leave the group of subjects out”, which consisted of 10 subjects who represented 15% of dataset in stratified manner. AU head was used as an auxiliary head which should have helped models to learn and generalize micro-expression relevant representations from the feature sets. It is also important to state that we have modified double emotion labels into a single ones, for example, ’positive+repression’ was mapped to ’repression’, those double labels are present only in 4DME and quite rare, around 7 samples per dataset, so we used them in order to improve the class balance. Only one rare category was left “sadness”, since there was not good mapping for it present, “fear” was moved to “others” since only 2 such samples were present in the whole data.

4.1 Sampling - Magnification - Optical Flow random split

4.1.1 3DCNN

Table 1: 3DCNN results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.47	105
Macro avg	0.38	0.38	0.38	105
Weighted avg	0.49	0.47	0.47	105

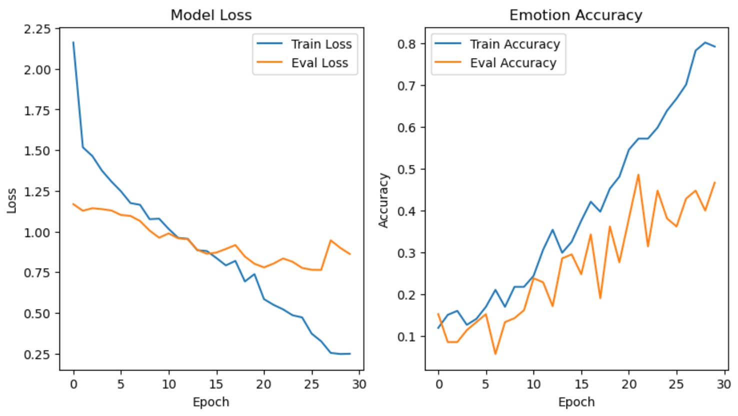

Figure 5: 3DCNN training graph

4.1.2 3DCONV + 2DCONV + biLSTM

Table 2: 3DCONV + 2DCONV + biLSTM results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.59	105
Macro avg	0.53	0.51	0.51	105
Weighted avg	0.59	0.59	0.58	105

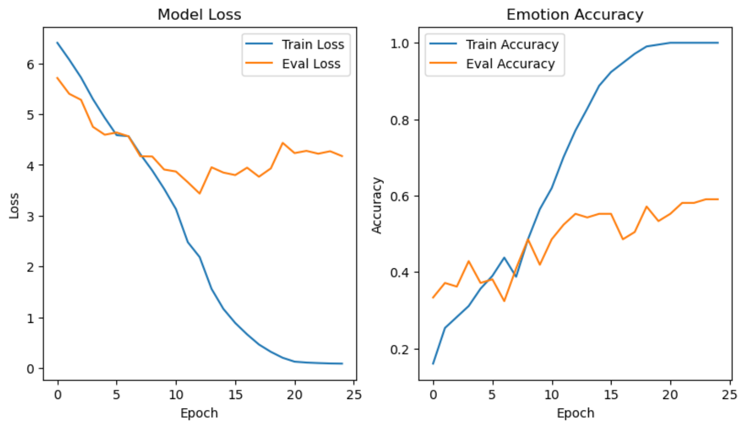

Figure 6: 3DCONV + 2DCONV + biLSTM training graph

4.2 Temporal Normalization - Optical Flow random split

4.2.1 3DCNN

Table 3: 3DCNN results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.66	105
Macro avg	0.68	0.66	0.63	105
Weighted avg	0.69	0.66	0.65	105

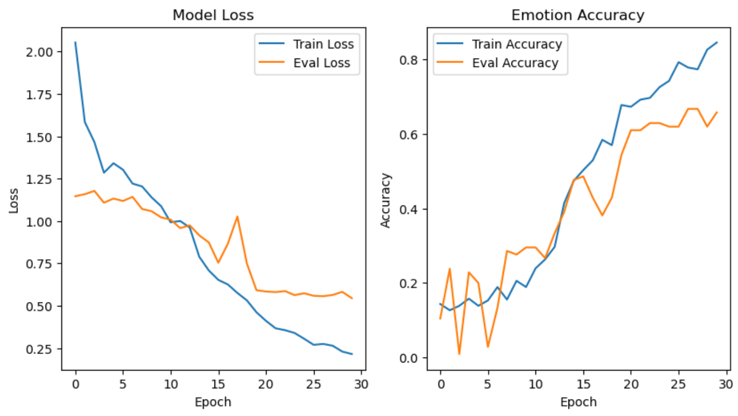

Figure 7: 3DCNN training graph

4.2.2 3DCONV + 2DCONV + biLSTM

Table 4: 3DCONV + 2DCONV + biLSTM results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.61	105
Macro avg	0.53	0.50	0.50	105
Weighted avg	0.61	0.61	0.60	105

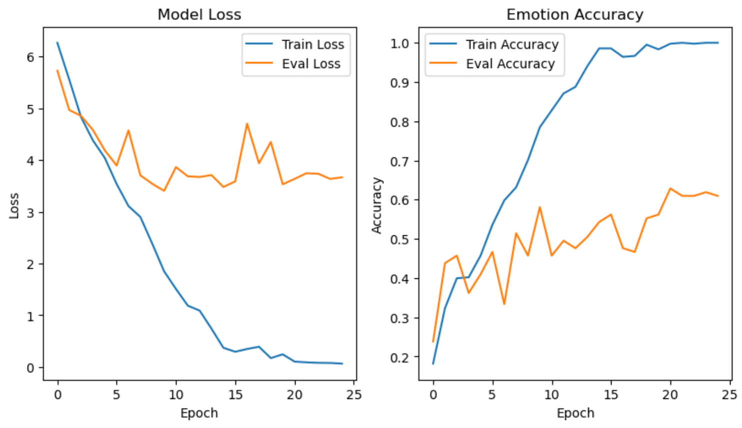

Figure 8: 3DCONV + 2DCONV + biLSTM training graph

4.3 Magnification – Temporal normalization random split

4.3.1 3DCNN

Table 5: 3DCNN results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.66	105
Macro avg	0.66	0.69	0.66	105
Weighted avg	0.68	0.66	0.65	105

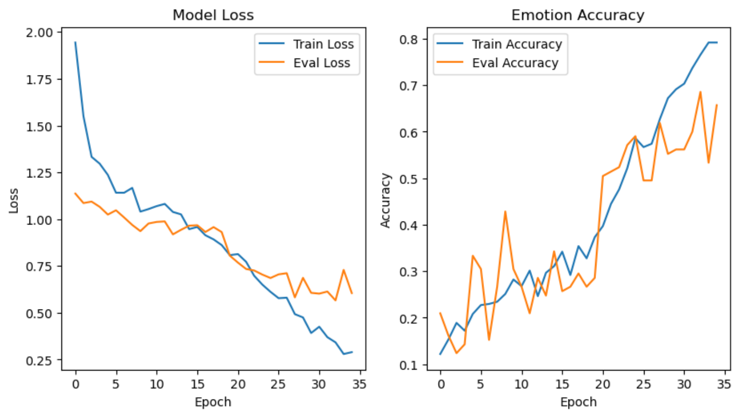

Figure 9: 3DCNN training graph

4.3.2 3DCONV + 2DCONV + biLSTM

Table 6: 3DCONV + 2DCONV + biLSTM results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.62	105
Macro avg	0.56	0.56	0.52	105
Weighted avg	0.61	0.62	0.59	105

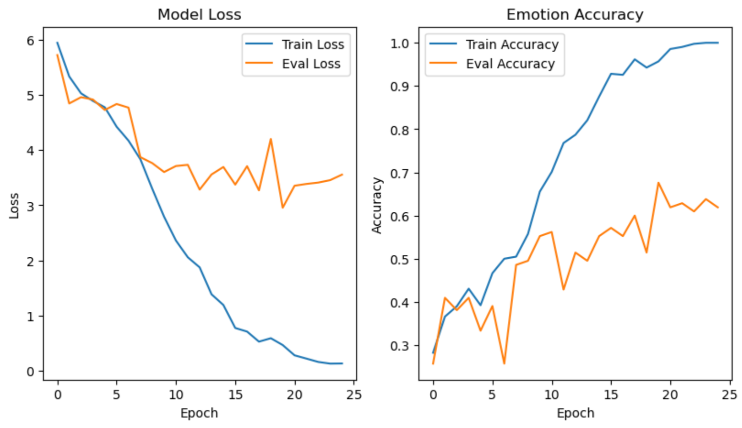

Figure 10: 3DCONV + 2DCONV + biLSTM training graph

4.4 Early fusion of Optical flow + LBP random split

LBPs were computed after magnification and sampling, Optical flow after directly after sampling.

4.4.1 3DCNN

Table 7: 3DCNN results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.58	105
Macro avg	0.58	0.55	0.52	105
Weighted avg	0.67	0.58	0.60	105

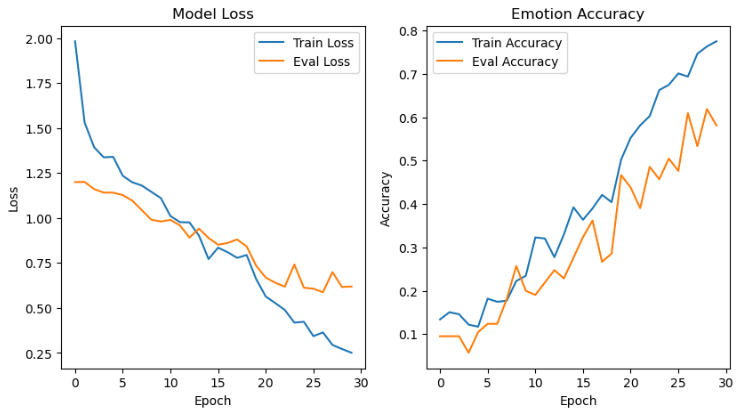

Figure 11: 3DCNN training graph

4.4.2 3DCONV + 2DCONV + biLSTM

Table 8: 3DCONV + 2DCONV + biLSTM results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.54	105
Macro avg	0.48	0.44	0.44	105
Weighted avg	0.54	0.54	0.53	105

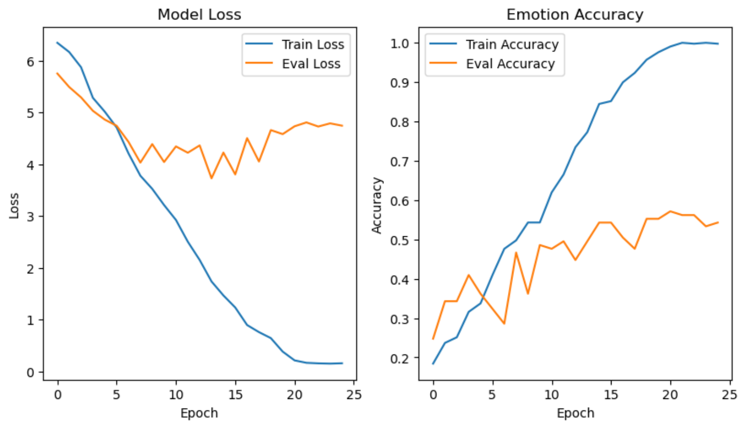

Figure 12: 3DCONV + 2DCONV + biLSTM training graph

Across all experiments of this branch, both architectures demonstrated pretty good ability to learn discriminative spatiotemporal representations from the feature sets, except set with optical flow which was computed from the motion magnified frames and variant of early fusion of LBPs with optical flow. It was noticed that first optical flow variant is very noisy since motion magnification magnifies a lot of "junk" with an upper bound cutoff 0.47. Bad results were also noticed in the early fusion of LBP with optical flow, that might have happened due to misaligned sequences, since framewise LBPs consisted of 8 frames and optical flow contained only 7, so LBPs were shifted in right by one frame. Nevertheless, the 3D CNN model, generally showed more stable and higher accuracy across most feature sets, achieving up to 0.66 accuracy and 0.68 weighted F1-score in the best-performing configuration. In comparison, the hybrid 3DCONV+ 2DCONV + BiLSTM model variant achieved competitive but slightly lower results, having a peak around 0.61 accuracy, depending on the feature representation. The 3D CNN's advantage likely arises from its fully spatiotemporal nature, which enables more effective capture of subtle motion cues and localized intensity changes, which are typical for micro-expressions. Across all conditions, F1-scores remained quite moderate 0.47–0.68, showing the inherent difficulty of micro-expression recognition. The action unit (AU) auxiliary head likely helps the networks form representations aligned with underlying muscle activations, even though AU classification was not directly evaluated. It is also important to notice how zigzagged evaluation loss of training graphs look like, in this case all learning rate reduction and scheduling callbacks were disabled, it is also important to state, that it was discovered that training around 30 epochs is the most optimal. In the results above are represented training using non-batched input, which might be the reason of such sharp changes in loss, since with batch normalization layers are unstable when batch sizes less than 4, our experiments also consider batch size of 4, as well with different learning rate scheduling mechanisms, which was looking more stable, but required up to 50-60 epochs, the results can be reviewed separately in the GitHub repository, but they have remained about the same. Since 2DCONV + biLSTM was consistently underperforming at this stage we stopped tuning this model further.

4.5 Motion magnification - Temporal Normalization - 3D FFT

In this case we present the result with batch size 4, and learning rate reduction by 10 times after 30 epochs.

Table 9: Motion magnification - temporal normalization - 3D FFT results

Metric	**Precision**	**Recall**	**F1-score**	**Support**
Accuracy	–	–	0.63	105
Macro avg	0.55	0.53	0.52	105
Weighted avg	0.63	0.63	0.62	105

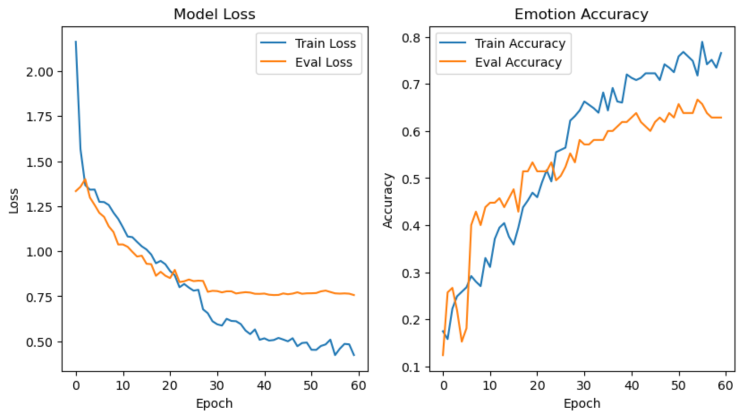

Figure 13: Motion magnification - Temporal Normalization - 3D FFT training graph

The model achieved an overall accuracy of 0.63 and a weighted F1-score of 0.62, which is consistent with previous high-performing feature sets such as Magnification - Temporal Normalization, and Temporal Normalization – Optical Flow. Noticeable that the training graph in this case looks much more stable due to bigger batch size and learning rate reduction policy, it was also discovered that models effectively learn until 80% of training accuracy, then happens sharp spike in evaluation loss, so it was necessary to keep this and other models around this level in future experiments for the generalization performance especially on future subject-disjoint evaluation.

4.6 3DCNN with "Leave the group of subjects out" or subject-disjoint evaluation

When we finally concluded our best model and best features, which appeared to be magnification of raw frames with consequent temporal normalization, and temporal normalization and consequent optical flow and strains extraction, it was decided to go further with model tuning and evaluation, and put the 3DCNN in the most difficult training-evaluation setup and evaluate the model performance on the subjects which model has never seen accidentally during the training. The functionality of our sampler was described in the preprocessing section, that is, sampler picks 10 subjects in stratified manner which form 15% of dataset, so that class balance in this testing sample remains the same.

4.6.1 Magnification – Temporal Normalization

Table 10: Magnification – temporal normalization results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.55	97
Macro avg	0.44	0.47	0.45	97
Weighted avg	0.54	0.55	0.54	97

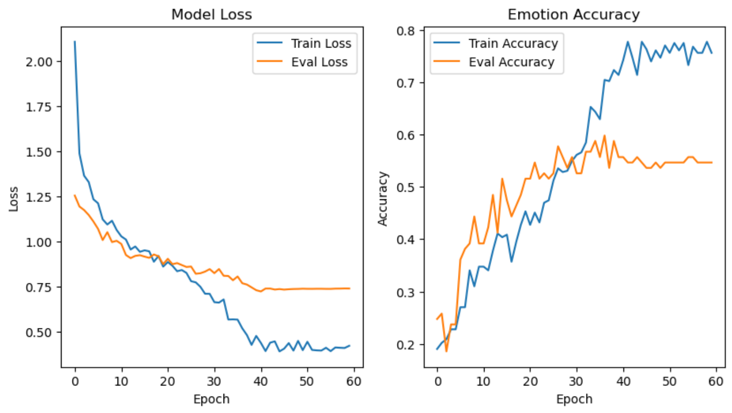

Figure 14: Magnification – Temporal Normalization training graph

Here in particular we have used exponential learning rate reduction after epoch 35, that is, learning rate was reduced twice every second epoch. This model also had slightly smaller spatial dropout (0.2) than the model for the next features.

4.6.2 Temporal Normalization – Optical Flow

Table 11: Temporal normalization – optical flow results

Metric	Precision	Recall	F1-score	Support
Accuracy	–	–	0.61	97
Macro avg	0.51	0.48	0.48	97
Weighted avg	0.62	0.61	0.59	97

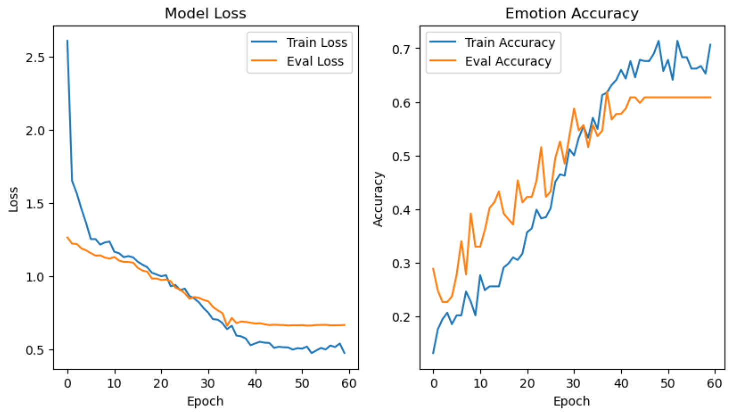

Figure 15: Temporal Normalization – Optical Flow training graph

The same exponential learning rate reduction policy was applied in this experiment, leading to the result, which can be considered state-of-the-art performance in micro-expression recognition given the limited size of the training corpus of 463 clips in total. It is important to state that batch size 1 often resulted in higher variance and "zigzaggy" evaluation graphs and worse results, but model trained on optical flow anyway stayed in 55% region, so we have switched to the setup presented above, as it was stated that batch normalization is very unstable when there is actually "no batch".

5 Discussion

In the beginning of the modelling and datasets manipulation there were a lot of mistakes done, which were further fixed, along with model development, and evolution of training-evaluation setups. In our project we reviewed 5 feature sets derived from the original images, and systematically compared different architectures, training and evaluation setups, as well as tuned and updated architectures numerous amounts of time, to build the model which is able to generalize on a such a limited corpus. Our subject subject-disjoint evaluation on best features which are optical flow vectors with strains, and motion magnified clips show that the 3D CNN achieved good generalization capacity even under the most extreme evaluation case. While performance decreased compared to random-split training as was expected, the accuracy of 0.61 and weighted F1-score of 0.59 is extremely high result on unseen individuals which is competitive with state-of-the-art micro-expression recognition benchmarks. The balanced, properly regularized 3D CNN benefits from motion-based features, although motion magnified frames showed also good results in the final evaluation.

Key things to mention is that data is extremely noisy, training results might fluctuate even if everything was kept as much deterministic as possible, it happens due to different gradient paths, different kernel initialization values and other hardcoded randomization aspects, which impact the results on unstable data.

It was noticed that Dropouts and Spatial Dropouts especially, are the key for the reduction of models focus on noise, Spatial Dropouts improved results approximately by 5-7%. It is also important to state that the best fitting to the training data which gives the best generalization results is around 80%, then happens sharp increase in evaluation loss. The biggest challenge is extremely small datasets, and heavily unbalanced labelling, so we tried to push the models size to the almost extreme minimum, while keeping their feature extraction power. Strong regularization, controlled model capacity, and balanced architecture depth are crucial for achieving good performance under data scarcity. Future experiments can potentially explore late fusion of 3D FFT and optical-flow representations, since both modalities capture complementary spatiotemporal frequency and motion information, which can open horizons for the new competitive achievements, also it is important to mention, that expanding the dataset by applying advanced data augmentation techniques can even further boost the performance of our models, even considering a small corpus which we currently have.

We have also developed an experimental transformer-based model but it remained untested due to time constraints, it can potentially give the results competitive to the 3DCNN when properly tuned.

6 Conclusion

The primary objectives of this project, which were stated as development of a robust deep learning architecture and evaluation of the impact of different feature families for facial micro-expression recognition were successfully achieved. Through tough systematic experimentation, the most representative and discriminative feature sets were identified, and a competitive 3D CNN-based recognizer was engineered, trained, and evaluated. The resulting model achieved 61% emotion classification accuracy on the subject-disjoint CASME II + 4DME split, placing it within the state-of-the-art range for a ME corpus of this size, interclass imbalance, and noise.

Apart from raw performance metrics and training graphs, the project has delivered a complete, end-to-end ME recognition pipeline, that is a clip preprocessing pipeline, exported feature banks, trained model weights, and a demo application capable of face detection and alignment via RetinaFace and real-time annotation of predicted emotions on raw video streams empowered by our models. This makes the system fully deployable and ready for the potential applications.

The project has gone far beyond course goals in time and space, this work integrates data analysis, research, state-of-art signal processing techniques, intensive deep learning modeling, and software engineering. The outcomes highlight that, in micro-expression recognition under limited-data conditions, careful preprocessing, temporal normalization, and strong regularization are far more important than network depth and number of trainable parameters.

7 Contribution distribution

Leo Davidov – 200+ hours. Preprocessing, feature engineering, modelling and tuning, project planning, presentation review and preparation, report writing.

Raffaele Sali – 50+ hours. Modelling and tuning, presentation review and preparation.

Sajjad Ghaeminejad – 35-40 hours. Face extraction, demo application.

Timofei Polishchuk – 15 hours. Writing the report.

Anatolii Fedorov – 10-15 hours. Writing the report, LaTeX editing.

Zhou Yang – Hours not specified. Writing the report.

References

[1] Weisong J Yan et al. "CASME II: An Improved Spontaneous Micro-Expression Database and the Baseline Evaluation". In: *PLOS ONE* 9.1 (Jan. 2014). DOI: 10.1371/journal.pone.0086041.

[2] Xiaobai Li et al. "4DME: A Spontaneous 4D Micro-Expression Dataset With Multimodalities". In: *IEEE Transactions on Affective Computing* 14.4 (2023), pp. 3031–3047. DOI: 10.1109/taffc.2022.3182342.

[3] Tuomas Varanka et al. "Data Leakage and Evaluation Issues in Micro-Expression Analysis". In: *arXiv preprint arXiv:2211.11425* (2022). Preprint. URL: https://arxiv.org/abs/2211.11425.

[4] Tuomas Varanka et al. *MEB: Micro-Expression Benchmark*. https://github.com/tvaranka/meb. Open-source library for micro-expression analysis. 2022.

[5] Elizabeth A Clark et al. "The Facial Action Coding System for Characterization of Human Affective Response to Consumer Product-Based Stimuli: A Systematic Review". In: *Frontiers in Psychology* 11 (May 2020), p. 920. DOI: 10.3389/fpsyg.2020.00920.

[6] Gunnar Farneback. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conference on Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370.

[7] Guoying Zhao and Matti Pietikainen. "Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 915–928.

[8] Yante Li. "Machine Learning for Perceiving Facial Micro-Expression". Ph.D. dissertation. Oulu, Finland: University of Oulu, 2022.

[9] Fang Wang et al. “A Dynamic 3D Spontaneous Micro-expression Database: Establishment and Evaluation”. In: *arXiv preprint arXiv:2108.00166* (2021).

[10] Xiaobai Li et al. “A Spontaneous Micro-expression Database: Inducement, Collection and Baseline”. In: *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. IEEE, 2013, pp. 1–6.

[11] Du Tran et al. “Learning Spatiotemporal Features with 3D Convolutional Networks”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 4489–4497. DOI: 10.1109/ICCV.2015.510.

[12] João Carreira and Andrew Zisserman. “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4724–4733. DOI: 10.1109/CVPR.2017.502.