ENHANCING SPEECH EMOTION RECOGNITION VIA FINE-TUNING PRE-TRAINED MODELS AND HYPER-PARAMETER OPTIMISATION

Aryan Golbaghi, Shuo Zhou

School of Computer Science, University of Sheffield, Sheffield, UK

{agolbaghi1, shuo.zhou}@sheffield.ac.uk

ABSTRACT

We propose a workflow for speech emotion recognition (SER) that combines pre-trained representations with automated hyperparameter optimisation (HPO). Using SpeechBrain's wav2vec2-base model fine-tuned on IEMOCAP as the encoder, we compare two HPO strategies, Gaussian Process Bayesian Optimisation (GP-BO) and Tree-structured Parzen Estimators (TPE), under an identical four-dimensional search space and 15-trial budget, with balanced class accuracy (BCA) on the German EmoDB corpus as the objective. All experiments run on 8 CPU cores and 32 GB RAM, without GPUs. GP-BO achieves 96.0% BCA in 11 minutes, and TPE (Hyperopt implementation) attains 97.0% in 15 minutes. In contrast, grid search requires 143 trials and 1,680 minutes to exceed 90% BCA, and the best AutoSpeech 2020 baseline reports only 85% in 30 minutes on GPU. For cross-lingual generalisation, an EmoDB-trained HPO-tuned model improves zero-shot accuracy by 25% on CREMA-D and 26% on RAVDESS. Results show that efficient HPO with pre-trained encoders delivers competitive SER on commodity CPUs.

***Index Terms*—** speech emotion recognition, pre-trained model, hyperparameter optimisation

1. INTRODUCTION

Speech emotion recognition (SER) plays a key role in human-computer interaction, mental health monitoring, and social robotics [1, 2]. Strong accuracy has typically relied on GPU resources and extensive manual tuning of hyperparameters. We target a low-cost, CPU-only pipeline that remains competitive with recent SER systems while requiring minimal expert intervention. To our knowledge, this is the first SER approach that combines a self-supervised backbone with automated machine learning (AutoML) to achieve state-of-the-art performance on commodity hardware.

Early SER systems relied on handcrafted acoustic and prosodic features such as MFCCs, pitch, and energy, paired with classical classifiers [3, 4]. In the 2010s, deep neural networks, including CNNs, RNNs, and hybrid architectures, became dominant by learning hierarchical time–frequency patterns directly from spectrograms [5, 6]. More recently, self-supervised pre-training has enabled general-purpose audio representations that can be fine-tuned for SER, with notable examples including wav2vec 2.0 [7], HuBERT [8], and ExHuBERT [9].

In parallel, automated machine learning (AutoML) techniques including hyperparameter optimisation (HPO) such as Gaussain Process Bayesian optimisation (GP-BO) [10] and tree-structured Parzen estimators (TPE) [11] have reduced the burden of manual tuning in many domains, but remain underexplored in SER. The AutoSpeech 2020 challenge [12] demonstrated the potential of automated pipelines for speech classification, though its reliance on GPU resources limited accessibility for low-cost or CPU-only deployments.

Despite these advances, three key challenges persist: (i) the high computational cost of model selection, (ii) the difficulty of optimising many interdependent hyperparameters, and (iii) limited cross-corpus and cross-language generalisability [13, 14]. HPO methods such as GP-BO and TPE offer a principled way to mitigate these issues by reducing manual search effort while preserving competitive accuracy.

2. ARCHITECTURE

We propose an AutoML fine-tuning workflow for SER, as demonstrated in Fig. 1. The inputs are: (i) a dataset split into training and validation sets, (ii) a pre-trained speech model, (iii) a k-dimensional hyperparameter vector, and (iv) an HPO algorithm.

2.1. Problem formulation

Let $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N_{\text{tr}}}$ and $\mathcal{D}_{\text{val}} = \{(x_j, y_j)\}_{j=1}^{N_{\text{val}}}$ denote training and validation sets. The encoder is initialised from a pre-trained network $f(\cdot; \theta_0)$. A hyperparameter vector $\lambda \in \Lambda \subset \mathbb{R}^k$ controls fine-tuning parameters such as learning rate, weight decay, scheduler settings, batch size, and dropout. For a given λ, parameters are obtained as

$$\theta^\star(\lambda) \leftarrow \text{Train}(f(\cdot; \theta_0), \lambda; \mathcal{D}_{\text{train}}). \tag{1}$$

Performance is measured by a validation metric M_{val} (balanced class accuracy, BCA), and the optimal configuration is

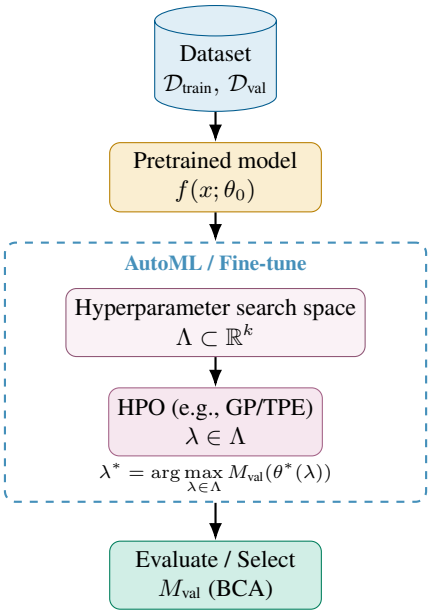

Fig. 1. Proposed AutoML fine-tuning workflow for SER. A pre-trained model is fine-tuned on $\mathcal{D}_{\text{train}}$, with HPO guiding hyperparameter search and M_{val} used for model selection.

selected as

$$\lambda^\star = \arg\max_{\lambda\in\Lambda} M_{\text{val}}(\theta^\star(\lambda)). \tag{2}$$

For a classification problem with C classes, BCA is

$$\text{BCA} = \frac{1}{C}\sum_{c=1}^{C} \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}, \tag{3}$$

where TP_c and FN_c are true positives and false negatives for class c.

2.2. HPO algorithms

We incorporate two families of sequential model-based optimisation (SMBO, Algorithm 1) as HPO candidates in our workflow and compare their performance (i) **Gaussian process Bayesian optimisation (GP-BO)** [10], and (ii) **Tree-structured Parzen estimators (TPE)** [11].

At iteration t, a surrogate is fit on history $\mathcal{D}_{t-1}$, candidate hyperparameters λ_t are selected by maximising an acquisition function, and the objective is evaluated as $y_t = f(\lambda_t)$. The history is then updated, and the process repeats.

GP-BO. Assume noisy scores $y = f(\lambda) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ and a GP prior on f. Conditioned on $\mathcal{D}_{t-1} = \{(\lambda_i, y_i)\}_{i=1}^{t-1}$, the predictive distribution is

$$f(\lambda) \mid \mathcal{D}_{t-1} \sim \mathcal{N}(\mu_t(\lambda), \sigma_t^2(\lambda)), \tag{4}$$

where μ_t is the GP mean and σ_t the predictive variance. Acquisition functions such as Expected Improvement (EI) [15] or GP-UCB [16] balance exploration and exploitation.

Algorithm 1 Sequential Model-Based Optimisation (SMBO)

Require: Search space Λ, budget T, initial design size n_0, acquisition $\alpha(\cdot)$

Initialize $\mathcal{D}_0 \leftarrow \{(\lambda_i, y_i)\}_{i=1}^{n_0}$ by evaluating $y_i = f(\lambda_i) + \varepsilon_i$ on a diverse design over Λ
$\lambda^\star \leftarrow \arg\min_{(\lambda,y)\in\mathcal{D}_0} y$ ▷ incumbent
for $t = 1$ to T **do**
  Fit surrogate s_t on $\mathcal{D}_{t-1}$ ▷ e.g., GP posterior or TPE densities
  $\lambda_t \leftarrow \arg\max_{\lambda\in\Lambda} \alpha(\lambda;\, s_t, \mathcal{D}_{t-1})$ ▷ maximize acquisition
  Evaluate $y_t \leftarrow f(\lambda_t) + \varepsilon_t$
  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\lambda_t, y_t)\}$
  $\lambda^\star \leftarrow \arg\min_{(\lambda,y)\in\mathcal{D}_t} y$
end for
return $\lambda^\star$ ▷ best configuration found

TPE. Instead of modelling $p(y|\lambda)$ directly, TPE estimates densities over configurations:

$$\ell(\lambda) = p(\lambda \mid y < y^\star), \quad g(\lambda) = p(\lambda \mid y \geq y^\star), \tag{5}$$

where $y^\star$ is a quantile threshold. Candidates are chosen by maximising $\ell(\lambda)/g(\lambda)$, which is proportional to expected improvement under this model.

3. EXPERIMENTS

3.1. Experimental setup

3.1.1. Dataset

We use the Berlin EmoDB corpus [17], a standard benchmark for SER containing 535 utterances from 10 speakers across 7 emotions. EmoDB has featured in community challenges such as AutoSpeech 2020 [12], making it suitable for comparison with published baselines. All recordings are converted to mono, resampled to 16 kHz, and either right-padded or truncated to a fixed length of T samples. Acoustic features are extracted on-the-fly using a pre-trained speech emotion encoder. Data are split via stratified partition, with the 80% for fine-tuning and the held-out 20% used once as the test set.

To assess cross-lingual generalisation, we additionally evaluate the EmoDB-trained model in a zero-shot setting on two English corpora: CREMA-D [18] and RAVDESS [19]. Since these datasets use different emotion taxonomies, we map their labels to a unified set aligned with EmoDB, as summarised in Table 1.

3.1.2. Backbone & classifier

Architecture. We adopt the `EncoderClassifier` module from SpeechBrain [20] with a wav2vec 2.0 encoder [7] pretrained in a self-supervised fashion and fine-tuned on

Dataset	Emotion
RAVDESS [19]	Angry, Disgust, Fearful, Happy, Neutral, Sad, Calm, Surprised
CREMA-D [18]	Anger, Disgust, Fear, Happy, Neutral, Sad
Mapped EMO-DB [17]	Anger, Disgust, Fear, Happiness, Neutral, Sadness, Boredom

Table 1. Emotion label sets in RAVDESS [19], CREMA-D [18], and the unified mapping aligned to EMO-DB [17]. The *"Surprised"* class from RAVDESS is excluded, as it is not present in EMO-DB, , which our model was fine-tuned on.

IEMOCAP [21] for categorical emotion. This checkpoint has not been exposed to EmoDB, preventing leakage and ensuring fair evaluation on the German target corpus. The encoder produces frame-level representations, which are temporally pooled into $\mathbf{z} \in \mathbb{R}^d$ and projected through a linear layer aligned with EmoDB's taxonomy:

$$\mathbf{z} = \mathrm{Pool}\big(f(\mathbf{x};\theta)\big), \tag{6}$$

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{z} + \mathbf{b} \in \mathbb{R}^C, \quad C = 7. \tag{7}$$

Training schedule. Training follows a two-stage strategy: the feature extractor is frozen for the initial epochs to stabilise learning on the small corpus, and then unfrozen at an HPO-controlled epoch $u \in \{0, \ldots, 5\}$ to enable task-specific adaptation once the classifier has converged. Optimisation uses cross-entropy loss, AdamW [22], and a cosine-annealed learning rate schedule.

3.1.3. HPO engines and search space

We compare GP-BO [10] and TPE [11], implemented in three AutoML engines: Ax [23] (for GP-BO), Hyperopt [24] (for TPE), and Optuna [25] (for TPE). All engines optimise the same compact four-dimensional space tailored to low-resource SER fine-tuning, which is summarised in Table 2.

GP-BO in Ax. Ax fits a Gaussian process surrogate over observed (λ, BCA) pairs and selects new configurations via an acquisition function (Expected Improvement by default). The search is seeded with a Sobol design and alternates surrogate fitting with acquisition maximisation. Ax also supports multi-point criteria (qEI/qUCB) and schedulers for early stopping and parallel evaluation.

TPE in Hyperopt. Hyperopt partitions past trials at a quantile threshold $y^\star$ into "good" and "bad" sets, fits kernel density estimates $\ell(x)$ and $g(x)$ to each, and samples new configurations by maximising $\ell(x)/g(x)$, proportional to expected improvement.

TPE in Optuna. Optuna allows programmatic search space definitions at runtime in TPE, supporting dynamic and hierarchical spaces. It further integrates asynchronous pruning

Hyperparameter	Space
`lr`	$[10^{-6}, 10^{-3}]$
`#epochs`	$\{1, \ldots, 10\}$
`unfreeze`	$\{0, \ldots, 5\}$
`maxlen`	{32k, 48k, 64k, 80k, 112k, 160k}

Table 2. Search space $\mathcal{H}$ for HPO experiments. `lr` is the learning rate. `#epochs` is the number of training epochs. `unfreeze` is the epoch index at which the pretrained encoder is unfrozen. `maxlen` is the per-utterance cap in samples (truncate/pad). We sample `lr` log-uniformly, `#epochs` and `unfreeze` uniformly over their ranges, and `maxlen` uniformly over its categorical set. Batch size is fixed to 1 to avoid LR–batch size interactions on the small corpus. Each engine is allocated 15 trials under identical seeds and training code. Learning rate is ; epochs and unfreeze are sampled uniformly over their discrete ranges; and max length is drawn uniformly from the categorical set.

with built-in strategies such as Median, Successive Halving [26, 27], and Hyperband [28]. We use Optuna's default TPE sampler in the experiments.

The selected HPO methods implemented in the three engines are compared against standard grid search, while the overall workflow is benchmarked against the best-performing solution from the AutoSpeech 2020 challenge [12].

3.1.4. Hardware setup

All HPO experiments were executed on a cluster with 8 CPU cores and 32 GB RAM; no GPU was used. The grid search baseline was run on a larger server with 64 CPU cores and 128 GB RAM. This "commodity" environment is the target deployment setting for our pipeline and was applied uniformly across Ax, Hyperopt, and Optuna so that the optimiser was the only varying factor.

For comparison, the best performed solution in AutoSpeech 2020 [12] entry on EmoDB reported results on a Tesla P100 GPU with 26 GB RAM, whereas our pipeline operates entirely on an 8-core CPU with 32 GB RAM. In raw compute throughput, the P100 is roughly 10–20$\times$ faster, but also about 5–10$\times$ more costly in resource terms.

3.2. Classification and efficiency performances

Table 3 reports threshold times and best-within-budget scores. Within a 15-trial budget, GP-BO (Ax) reached both > 0.80 and > 0.90 BCA fastest (2.4 min, trial 2), while TPE (Hyperopt) achieved the highest overall BCA (0.97 in 15 min). TPE (Optuna) also exceeded 0.90 BCA but required substantially longer (185 min). The corresponding hyperparameter settings (Table 4) show convergence to similar learning rates and sequence lengths, with differences mainly in unfreeze epoch

Method	$>$ **0.80 BCA**	$>$ **0.90 BCA**
Grid search	1056 min (trial 101)	1680 min (trial 143)
TPE (Optuna)	7.8 min (trial 4)	185.3 min (trial 15)
TPE (Hyperopt)	6.4 min (trial 5)	15 min (trial 11)
GP-BO (Ax)	**2.4 min (trial 2)**	**2.4 min (trial 2)**

Table 3. Comparison of time to reach accuracy thresholds across different HPO engines. BCA denotes balanced class accuracy; TPE refers to tree-structured Parzen estimators [11]; and GP-BO denotes Gaussian process Bayesian optimisation [10]. Optuna [25], Hyperopt [24], and Ax [23] are engines (implementations) of these methods. The best result is highlighted in **bold**.

Method (engine)	`lr`	`#Ep.`	`unfrz`	`maxlen`	**BCA**	**Duration**
TPE (Optuna)	8.89e-06	4	1	64k	0.93	185 min
TPE (Hyperopt)	2.59e-05	8	4	80k	0.97	15 min
GP-BO (Ax)	2.28e-05	9	0	80K	0.96	11 min

Table 4. Best hyperparameters and performance obtained by the three AutoML engines (Ax, Hyperopt, Optuna) within 15 trials. `#Ep.` = number of epochs; `unfrz` = encoder unfreeze epoch; `maxlen` = input length (samples).

and number of training epochs. To quantify speed–accuracy trade-offs, we define an efficiency metric

$$\mathcal{E} = \frac{\text{BCA}}{\text{wall-clock minutes to reach that BCA}}, \qquad (8)$$

with results summarised in Table 5. Despite running entirely on an 8-core CPU (no GPU), GP-BO (Ax) delivered the best balance of speed (11 min) and accuracy (0.96 BCA), while Hyperopt achieved slightly higher BCA (0.97) at modest extra cost (15 min). Both approaches substantially outperform the best AutoSpeech 2020 EmoDB solution (0.85 BCA in $\sim$30 min on GPU), yielding a $\sim 3\times$ efficiency improvement. Moreover, our fine-tuned model attained 98.3% raw accuracy on EmoDB, surpassing the reported performance of native German listeners (84%) [29].

In summary, GP-BO (Ax) provides the best speed–accuracy trade-off for rapid iteration, while TPE (Hyperopt) achieves the top accuracy with only slightly higher time cost, establishing our CPU-only workflow as both more efficient and more accurate than prior GPU-based baselines.

3.3. Zero-shot transfer across corpora

Table 6 reports that the pre-trained model performs poorly in zero-shot settings (without fine-tuning), with BCA close to chance on EmoDB (0.11), CREMA-D (0.13) and RAVDESS (0.18). In contrast, HPO-tuning on EmoDB (using GP-BO in Ax) substantially improves cross-lingual generalisation, raising BCA to 0.38 on CREMA-D and 0.44 on RAVDESS. On the source corpus EmoDB, BCA increased to 0.96 after tuning, confirming the effectiveness of the proposed workflow.

Framework	**Best BCA**	**Time to Best**	$\mathcal{E}_{\text{best}} \uparrow$
Grid Search	0.98	1680 min	0.0005
AutoSpeech 2020 [12]	0.85	30 min	0.0280
TPE (Optuna)	0.93	185 min	0.0050
TPE (Hyperopt)	0.97	15 min	0.0646
GP-BO (AX)	**0.96**	**11 min**	**0.0872**

Table 5. Best balanced class accuracy (BCA), time to reach it, and efficiency ($\mathcal{E}_{\text{best}}$) across search methods. GP-BO (Ax) provides the best trade-off between accuracy and time, while TPE (Hyperopt) achieves the highest peak BCA.

Corpus	**BCA (Zero-shot)**	**BCA (HPO-tuned)**
EmoDB	0.11	0.96
CREMA-D	0.13	0.38
RAVDESS	0.18	0.44

Table 6. Zero-shot vs. HPO-tuned performance. The HPO-tuned performance on the two English corpora CREMA-D and RAVDESS are obtained by a model trained with EmoDB with the hypermaters from the Ax engine. HPO-tuning yields large gains both in-domain and for cross-lingual transfer.

4. CONCLUSION

We proposed a CPU-only SER workflow that combines a pre-trained wav2vec 2.0 encoder with automated hyperparameter optimisation (HPO). Two HPO methods, GP-BO and TPE, were evaluated through three engines: Ax, Hyperopt, and Optuna. On the German SER corpus EmoDB, Ax and Hyperopt achieved over 96% balanced class accuracy within minutes, outperforming the best AutoSpeech 2020 solution (85% in 30 min on GPU) while avoiding the $100\times$ cost of grid search. HPO-tuning also improved cross-lingual transfer, increasing the SER accuracy of an EmoDB-trained model by 25% on CREMA-D and 26% on RAVDESS compared to the baseline.

These results demonstrate that pre-trained encoders coupled with efficient HPO can deliver state-of-the-art SER performance without GPUs. Future work will extend to multilingual corpora, explore parameter-efficient fine-tuning, and integrate energy and latency objectives for deployment in resource-constrained settings.

5. REFERENCES

[1] Moataz El Ayadi, Mohamed S Kamel, and Fakhri Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognition*, vol. 44, no. 3, pp. 572–587, 2011.

[2] Björn W Schuller, "Speech emotion recognition: Two decades in a nutshell, benchmarks, and ongoing trends," *Communications of the ACM*, vol. 61, no. 5, pp. 90–99, 2018.

[3] Björn Schuller, Stefan Steidl, and Anton Batliner, "The interspeech 2009 emotion challenge," in *Interspeech*, 2009.

[4] Florian Eyben et al., "The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing," *IEEE Transactions on Affective Computing*, vol. 7, no. 2, pp. 190–202, 2015.

[5] Kun Han, Dong Yu, and Ivan Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Interspeech*, 2014.

[6] George Trigeorgis et al., "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *ICASSP*. IEEE, 2016, pp. 5200–5204.

[7] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *NeurIPS*, 2020, vol. 33, pp. 12449–12460.

[8] Wei-Ning Hsu et al., "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[9] Shahin Amiriparian, Filip Packań, Maurice Gerczuk, and Björn W. Schuller, "Exhubert: Enhancing hubert through block extension and fine-tuning on 37 emotion datasets," in *Interspeech*, 2024, pp. 2635–2639.

[10] Jasper Snoek, Hugo Larochelle, and Ryan P Adams, "Practical bayesian optimization of machine learning algorithms," in *NeurIPS*, 2012, vol. 25.

[11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl, "Algorithms for hyper-parameter optimization," in *NeurIPS*, 2011, vol. 24.

[12] Jingsong Wang et al., "Autospeech 2020: The second automated machine learning challenge for classification," in *Interspeech*, 2020.

[13] Björn Schuller et al., "Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge," *Speech Communication*, vol. 53, no. 9-10, pp. 1062–1087, 2011.

[14] Taiba Majid Wani et al., "A comprehensive review of speech emotion recognition systems," *IEEE Access*, vol. 9, pp. 47795–47814, 2021.

[15] Donald R Jones, Matthias Schonlau, and William J Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[16] Niranjan Srinivas et al., "Gaussian process optimization in the bandit setting: No regret and experimental design," in *ICML*, 2010.

[17] Felix Burkhardt et al., "A database of german emotional speech.," in *Interspeech*, 2005, vol. 5, pp. 1517–1520.

[18] Houwei Cao et al., "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE Transactions on Affective Computing*, vol. 5, no. 4, pp. 377–390, 2014.

[19] Steven R Livingstone and Frank A Russo, "The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english," *PloS one*, vol. 13, no. 5, pp. e0196391, 2018.

[20] Mirco Ravanelli et al., "Open-source conversational ai with speechbrain 1.0," *JMLR*, vol. 25, no. 333, 2024.

[21] Carlos Busso et al., "Iemocap: Interactive emotional dyadic motion capture database," *Language Resources and Evaluation*, vol. 42, no. 4, pp. 335–359, 2008.

[22] Ilya Loshchilov and Frank Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[23] Miles Olson et al., "Ax: a platform for adaptive experimentation," in *AutoML 2025 ABCD Track*, 2025.

[24] James Bergstra, Daniel Yamins, and David Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *ICML*. PMLR, 2013, pp. 115–123.

[25] Takuya Akiba et al., "Optuna: A next-generation hyperparameter optimization framework," in *KDD*, 2019.

[26] Zohar Karnin, Tomer Koren, and Oren Somekh, "Almost optimal exploration in multi-armed bandits," in *ICML*. PMLR, 2013, pp. 1238–1246.

[27] Kevin Jamieson and Ameet Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *AISTATS*. PMLR, 2016, pp. 240–248.

[28] Lisha Li et al., "Hyperband: A novel bandit-based approach to hyperparameter optimization," *JMLR*, vol. 18, no. 185, pp. 1–52, 2018.

[29] Sefik Emre Eskimez et al., "Emotion classification: how does an automated system compare to naive human coders?," in *ICASSP*. IEEE, 2016, pp. 2274–2278.