Prediction of Valence from Symbol-based Text Input Considering Predictive Uncertainty using Deep Learning

Nicolas Kolbenschlag
University of Augsburg
Augsburg, Germany
nicolas.kolbenschlag@student.uni-augsburg.de
Supervisor: Lukas Stappen, M.Sc.

ABSTRACT

Estimating and reducing uncertainty is essential when using neural networks. Especially in Emotion Recognition, because there often is no solid *ground-truth*, as an emotional state mostly can not be valued objectively.

The overall purpose of this study is to investigate several techniques to deal with predictive uncertainty. Especially in the field of emotions recognition from text-based input features, processed by large transformer models like BERT [6].

This work introduces some common approaches to reduce uncertainty in deep neural networks and applies them to the baseline model for predicting valence or arousal from the MuSe 2020 challenge. The MuSe-CaR dataset, which comes with the challenge, contains annotations from multiple annotators. Emotion is rated from a very objective point of view. This circumstance raises the question, whether the uncertainty of a deep learning model, trained on that data, would be high and how the model could be made more robust against it. Therefore, we adapt the baseline model to these techniques and compare the results.

This showed that Quantile Regression, as it is based on the Mean-Absolute-Error, needs to change to loss optimization landscape, whereby it is not suitable for the given problem state.

Furthermore, we propose to transfer the idea of tilting the loss (during training of a neural network) at several certain points for different output nodes, used along with the Mean-Absolute-Error for Quantile Regression, to other loss functions. This allows us to make use of the beneficial effects of Quantile Regression on predictive uncertainty, without refraining from using an appropriate loss function for a given problem state.

Keywords

Deep Neural Network, Uncertainty, Emotion, Natural language processing

1. INTRODUCTION

Uncertainty-awareness is a large topic among researchers. Without any technique for measuring or avoiding predictive uncertainty, the output of a neural net has to be accepted, whether the model was able to make a meaningful prediction from an input sample or not. And if not, there would be no way to determine.

First, we need to break down what circumstances can cause predictive uncertainty. Bachstein names two main kinds of uncertainty, *aleatory* and *epistemic* uncertainty [3]. Aleatory uncertainty is caused by the non-deterministic of an estimated problem, which leads to different output for similar inputs (e.g. the landing point of an arrow). Epistemic uncertainty describes the fact that a neural network produces output on a given input sample, whether it had been trained on comparable samples or not (e.g. image of a ship fed to a cat-vs.-dog classifier).

In this work, we want to investigate approaches to deal with such predictive uncertainty. Some approaches try to picture (so modelling) and measure uncertainty (compare figure 5), while others might try to avoid it. We have to keep in mind that different types, so causes, of uncertainty require different ways of handling them. As one can think about applies that epistemic uncertainty sometimes probably can not be avoided, it just could be measured. For example an image of a ship fed a cat-vs.-dog classifier should never predict neither cat nor dog with any certainty. But the model should output a low certainty, which could be interpreted as a signal that none of the possible classes is accurate. Aleatory uncertainty on the other hand could possibly be decreased.

A naive approach when training a model for classification would be to use the relative strength of an output node for the degree of (predicted) uncertainty. But this doesn't work for regression problems and only could possibly be used to measure epistemic uncertainty, but not for aleatory uncertainty.

We applied several strategies to the MuSe-Car dataset and compared their impacts on predictive uncertainty.

In the following sections, we give a short overview over some approaches to handle predictive uncertainty. Afterwards we present the methods that we used in our experiments. At the end we show our results and explain some conclusions that can be drawn from the observations.

1.1 Related work

The following sections briefly introduce some approaches to deal with predictive uncertainty of neural networks.

1.1.1 *Approaches to handle predictive uncertainty*

Bayesian neural networks combine neural networks and stochastic modelling. Instead of generating standalone predictions, they output probabilistic assurances on their prediction [14]. Blundell et. al. propose an algorithm for learning probability distribution parameter as neural network weights [4]. In classical neural networks, a weights consists of a fixed single value, which leads to deterministic inference. But if a weights consists of probabilistic distributional parameter, inference equals to calling a probabilistic distribution, which makes it non-deterministic.

The probably most common way to deal with a network's predictive uncertainty is called *Monte Carlo Dropout* [7]. For that, dropout is used at prediction time, instead of just at training time. *Dropout ensembles* is the calculation of distributional parameter from multiple forward runs with Monte Carlo Dropout.

Another very simple, but computational expensive, approach is *Ensemble averaging*, in which several neural networks (different, but with the same architecture) are trained from varying starting points (seeds) and distributional parameter are calculated from their predictions [3].

Both Monte Carlo Dropout and Ensemble averaging, try to avoid aleatory uncertainty, as unwanted differences among forward runs (predictions) get averaged out.

Adversarial training is a method to improve the robustness of a neural network. During adversarial training, given an input sample consisting of features and a label, an additional sample with features similar, but different, to the original features, is created and also feed to the network [23]. Combined with a proper scoring rule (loss function) and the usage of multiple predictions runs (ensembles), adversarial training can also lead to more uncertainty-awareness [13].

Another approach is *data augmentation*, which also is a widely used technique to improve a model's robustness. Here the number of input samples gets increased through variations of original training samples. It is hoped that these variations also improve the model's robustness. This idea is similar to adversarial training, as in both cases, the input features are modified in some way.

Quantile regression is an approach that estimates quantiles. Each estimated quantile τ_j (e.g. 0.1, 0.5 and 0.9) is estimated by an associated output node [12]. The loss for each quantile / node is described by

$$\mathcal{L}_{tilted_j} = max(\tau_j * e_j, (\tau_j - 1) * e_j), \qquad (1)$$

with the Mean-Absolute-Error e. This loss function is called *tilted loss*, because it tips the error by the affiliation to the given quantile.

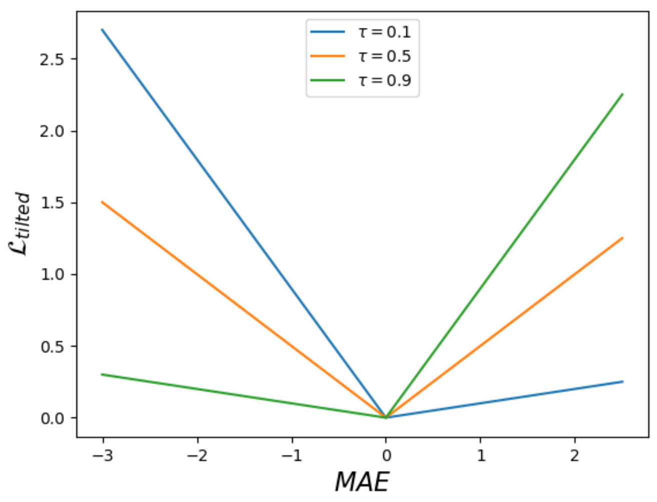

Figure 1: Tilted loss function for the 0.1**,** 0.5 **and** 0.9 **quantile**

During inference the model takes the mean of all quantiles.

An application comes from Stappen et. al.[19]. They used *soft labels* [2] and quantile regression to decrease uncertainty in modelling of peer reviews for academic papers, which is a highly objective task, as multiple reviewers could and often do have very different opinions on a submission.

1.1.2 *Distributional parameter estimation*

The previous approaches have in common that they produce an output, that can be used to calculate distributional parameter, like mean (prediction) and variance (uncertainty). Instead of calculating these parameters indirectly, a neural network can be trained to learn a direct mapping from input data to its own uncertainty. Thus, the network has an additional output node for the uncertainty.

Han et. al. [9] propose a framework where a neural network is trained to model the uncertainty in addition to the actual prediction by creating an embedding of input features in a (shared) embedding layer, followed by two separated forward paths. One for predicting the actual prediction, namely a *gold-standard* of multiple annotations, computed with the Evaluator Weighted Estimator (EWE), which is described in [8]. And the other for measuring the uncertainty, which is defined as the standard deviation among the multiple annotations. Notice that this approach requires, that the dataset contains multiple annotations, which is not always given.

Another very notable approach comes from Wu et. al. [22]. They introduce a framework for learning the uncertainty of contrastive learning representation, by mapping a representation (from a pretrained encoder as usual in contrastive learning [11]) to a distribution in representation space, whose variance is taken as measurement for uncertainty.

1.1.3 *Uncertainty in dataset labels*

This approach by Northcutt et. al. [16] focuses on cleaning the dataset, by finding label errors and noisy data, so that predictive uncertainty automatically decreases. Instead of trying to deal with challenging samples, that would increase uncertainty (like the above approaches), such samples are, to put it simply, removed (or re-weighted) from the dataset.

Natarajan et. al. describe weighting the loss function by the noise of a sample in [15].

2. METHODOLOGY

In the following sections we give an overview about the techniques used in our experiments. We look at our loss functions, uncertainty-methods and the metrics you need to know when following the experimental results. At all experimented with three techniques to decrease predictive uncertainty: Monte Carlo Dropout, Quantile Regression and re-weighting samples during training according to their expected uncertainty.

2.1 Performance metrics

For performance measurement, we utilized the official evaluation metric of the MuSe 2020 challenge [18], namely *Concordance Correlation Coefficient (CCC)*, which is often used in similar challenges[21]. It is described by:

$$CCC = \frac{2r_{x,y}\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}, \quad (2)$$

where $r_{x,y}$ denotes the *Pearson correlation coefficient* between two signals (e.g. label/gold-standard and prediction), μ_x and μ_y the means and σ_x and σ_y the standard deviations of those two signals.

The CCC ranges for -1 to 1 and higher values indicate higher concordance [9]. For us a high CCC score indicates good predictive performance.

As the samples in the MuSe-CaR dataset are annotated continuously, the choice of a metric that takes this characteristic in account is necessary. Therefore we used the CCC with its correlation computation. To have a more "classical" metric, that measures the performance at each time step separated, as comparison, we also computed the *Root-Mean-Squared-Error (RMSE)*, defined by:

$$RMSE = \sqrt{\frac{1}{N} * \sum_{i=1}^{N}(\hat{y_i} - y_i)^2}, \quad (3)$$

where $\hat{y_i}$ denotes the predicted values, y_i the label and N the batch size.

The RMSE measures the distance from the predictions to the labels, that is why a lower RMSE score indicated better model performance.

2.2 Uncertainty metrics

As the CCC is our main performance metric, we also use it to measure a technique's uncertainty. We repeated all of our experiments for 10 iterations starting from different seeds. Afterwords we computed mean μ, standard deviation σ, minimum and maximum from the 10 resulting CCC scores.

$$\mu = \frac{\sum_i x_i}{N} \quad (4)$$

$$\sigma = \sqrt{\frac{\sum_i |x_i - \mu|^2}{N}} \quad (5)$$

x_i denotes the i-th value and N is the number of samples.

We expect the standard deviation to be low, if the models predictive uncertainty is also low. And we expect it to be high, if the uncertainty is high. The reason for the assumption is, that if the model was not able to learn the problem statement with a certain certainty, every seed should converge to different states, which would lead to varying predictions (among seeds).

2.3 Basic training criterions

For fitting the baseline model, as described above, we used two different loss functions and compared the results.

First, we used the CCC. Because larger CCC scores indicate good performance, it has to be contrary to be used for gradient descent.

$$\mathcal{L}_{CCC} = 1 - CCC \quad (6)$$

We compared the results with $\mathcal{L}_{CCC}$ to training with the *Mean-Absolue-Error (MAE)*:

$$\mathcal{L}_{MAE} = \frac{1}{N} * \sum_{i=1}^{N} |\hat{y_i} - y_i|, \quad (7)$$

where $\hat{y_i}$ denotes the predicted values, y_i the label and N the batch size. We decided to use the MAE, especially because the loss function for Quantile Regression ($\mathcal{L}_{tilted}$) is based on it and we can use it as benchmark when evaluating the experimental results.

2.4 Correlation-weighted CCC (cwCCC)

In addition, we tried to decrease predictive uncertainty by re-weighting the loss for examples during training. As mentioned, the MuSe-CaR datset contains multiple annotations (which were used to compute the gold-standard). By computing the *Pearson correlation coefficient* $r_{a,b}$ of two annotators a and b for each sample, we got a measurement of uncertainty "transported" with a given sample denoted with c_i, by averaging all pairs of annotations:

$$c_i = \frac{\sum_{a,b \in A_i} r_{a,b}}{\sum_{j=1}^{|A_i|-1} j}, \quad (8)$$

with the annotations A_i for the given example i.

We apply a possibility to use this information to decrease predictive uncertainty and increase performance of the model. It is based on the idea of re-weighting the CCC loss $\mathcal{L}_{CCC_i}$ for sample i[1]:

$$\mathcal{L}_{cwCCC_i} = \frac{c_i + 1}{2} * \mathcal{L}_{CCC_i} \quad (9)$$

The underlying reasoning is that the loss for a sample, that transports high uncertainty, becomes weakened. The model is forced to disregard samples that are hard to learn and focuses on those that should be predictable with some certainty.

3. EXPERIMENTS

3.1 Project environment

[1]As the Pearson correlation coefficient has range [-1,1], we scale c_i to [0,1].

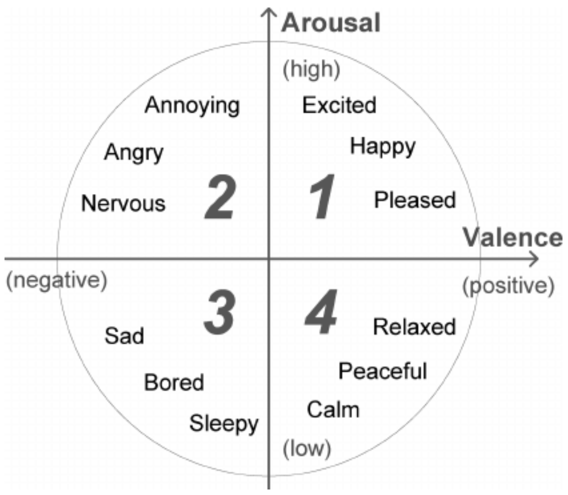

Figure 2: Emotions represented in a 2-dimensional space, from [24].

3.1.1 *The MuSe-CaR dataset*

For Each of the following experiments, we utilized the transcriptions of the MuSe-CaR datset[18]. MuSe-CaR is a large and multi-modal (language transcriptions, audio and video) annotated dataset. The samples were sourced from YouTube and contain car reviews with English speakers.

The annotators assume that can emotions be represented in a 2-dimensional space, as proposed by Russel [17]. In it 2 features, namely *valence* and *arousal*, are assigned to an emotional-state as shown in figure 2. While the sentiment of the emotion is expressed by valence, arousal corresponds in this model to the excitement.

The examples were annotated continuously (one label per time step) by three dimensions, as there are valence, arousal and *trustworthiness* [1] by multiple annotators. Afterward a *gold-standard* was calculated, using the *Evaluator Weighted Estimator (EWE)* [8]. The EWE generates a kind of a *mean series* from multiple individual series, but with outliers ignored, so that they don't distort the overall picture (annotation).

As it was the initial purpose of this work, to investigate uncertainty with natural language processing models like BERT [6], we restricted to the transcriptions. Also, to keep things clear, we limited ourselves on valence as prediction target.

3.1.2 *Feature extraction*

The first part of the model pipeline that calculates a label from a raw text transcription is an embedding technique. We used BERT [6] to generate encodings from the transcriptions, which then could be fed into the actual prediction model, by feeding them to the BERT-model and storing the output of the last 4 layers (*bert-4*). We also used *fastText* [5] to generate the features differently to have a comparison to the bert-4 features.

3.1.3 *Baseline model*

The actual regression model, is defined as follows: For our experiments we have built upon the *Multi-modal Continuous Dimensional Emotion Recognition Using LSTM and Self-Attention Mechanism* [20]. The model (with our configuration) consists of one self-attention layer with 4 heads, followed by one bidirectional-LSTM layer with 64 hidden states and closed with one linear fully-connected output layer with also 64 hidden states as you can see in figure 3.

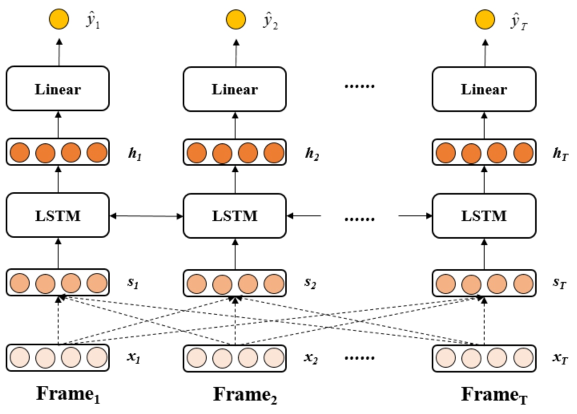

Figure 3: Our baseline model: Multi-modal Continuous Dimensional Emotion Recognition Using LSTM and Self-Attention Mechanism [20] (the dotted lines denote the self-attention mechanism)

3.2 Experimental settings

3.2.1 *Monte Carlo Dropout (MCD)*

When adapting the baseline model for Monte Carlo Dropout, we added Dropout at training and test (prediction) time with a dropout rate of 0.3 to the self-attention and the output layer. For inference, we averaged the output of 10 forward runs.

3.2.2 *Quantile Regression*

For using Quantile Regression we changed the output dimension of the final layer from 1 to 3 (one for each quantile, namely 0.1, 0.5 and 0.9) and applied the $\mathcal{L}_{tilted_j}$ to each of them. This results in the following training criterion:

$$\begin{aligned}\mathcal{L}_{tilted} &= \frac{1}{3} * \sum_{i=1}^{3} \mathcal{L}_{tilted_i} \\ &= \frac{1}{3} * \sum_{i=1}^{3} max(\tau_i * e_i, (\tau_i - 1) * e_i),\end{aligned} \tag{10}$$

with the Mean-Absolute-Error e as mentioned above and the quantiles $\tau = [0.1, 0.5, 0.9]$.

4. RESULTS

In the following sections we share the observations from our experiments. The sections describe the most remarkable findings derived from our observations.

4.1 An appropriate loss function to avoid uncertainty

If you look at table 1 you see the CCC score of seeds trained with losses based on $\mathcal{L}_{MAE}$. As we use the standard deviation of CCC scores among seeds as measurement for uncertainty of a model, we would expect the standard deviation to be lower with Monte Carlo Dropout or Quantile

	$\mathcal{L}_{MAE}$		$\mathcal{L}_{MAE}$ + MCD		$\mathcal{L}_{tilted}$	
seed	test	dev	test	dev	test	dev
314	.2721	.2112	.0097	.0417	.4648	.3465
315	.2578	.2667	.4123	.3427	.0027	.0090
316	.0072	.0413	.0051	.0317	.0056	.0187
317	.4659	.3692	.0030	.0061	.0006	.0067
318	.0497	.0978	.0047	.0279	.4425	.3731
319	.3066	.2986	.4495	.3175	.4329	.3520
320	.0871	.1734	.4160	.3663	.4512	.3611
321	.0087	.0423	.0060	.0389	.4048	.3372
322	.0140	.0488	.4608	.3278	.0007	.0036
323	.3960	.3226	.0641	.0671	.0042	.0092
Mean	.1865	.1872	.1831	.1568	**.2210**	.1817
Std.	.1735	.1249	.2177	.1577	.2305	.1819
Min.	.0072	.0413	.0030	.0061	.0006	.0036
Max.	.4659	.3692	.4608	.3663	.4648	.3731

Table 1: CCC scores for experiments with bert-4 based on $\mathcal{L}_{MAE}$

	$\mathcal{L}_{MAE}$		$\mathcal{L}_{MAE}$ + MCD		$\mathcal{L}_{tilted}$	
seed	test	dev	test	dev	test	dev
314	.4314	.3998	.4424	.4033	.3775	.3872
315	.4458	.4076	.3842	.3848	.4417	.4071
316	.4416	.4077	.4417	.4051	.4403	.4080
317	.3843	.3809	.4455	.4196	.4431	.4305
318	.4510	.4270	.4408	.4104	.3810	.3895
319	.3967	.4235	.3750	.4064	.3760	.3960
320	.4794	.4031	.3852	.4040	.3773	.3887
321	.4417	.4085	.4427	.4030	.3950	.3880
322	.4451	.4050	.3761	.3844	.4409	.4077
323	.3951	.3958	.4473	.4115	.4401	.4066
Mean	.4312	.4059	.4181	.4033	.4113	.4009
Std.	.0299	.0131	.0329	.0110	.0320	.0137
Min.	.3843	.3809	.3750	.3844	.3760	.3872
Max.	.4794	.4270	.4473	.4196	.4431	.4305

Table 2: RMSE scores for experiments with bert-4 based on $\mathcal{L}_{MAE}$

Loss than with usual $\mathcal{L}_{MAE}$, but the opposite is the case. And the standard deviations are extremely high compared to models trained on losses based on $\mathcal{L}_{CCC}$ (table 4). The losses vary from around zero to 0.4658. There has to be another reason, than just uncertainty, therefore. If we now, look at the RMSE scores for that models in table 2, we observe that a better (higher) CCC score does not necessarily cause a better (lower) RMSE score. For example seed 316 performs better than seed 318 in RMSE, but worse in CCC (trained with $\mathcal{L}_{MAE}$).

To understand these unexpected breakdowns of seeds when trained with $\mathcal{L}_{MAE}$ based loss functions, we have to look at the following chart in figure 4.

The golden graph in figure 4 is a randomly generated signal, the blue one is the golden one moved up by 0.35 and the red one is the mean of the golden. Following from this the MAE for the red graph is 0.23 and the CCC is 0. For blue the MAE is 0.35 and the CCC 0.54. The crucial point is that the red line fits the golden better than the blue in terms of MAE, although it doesn't fit in terms of CCC at all.

This shows that predicting the mean might be a local minimum in the MAE optimization landscape. This is what

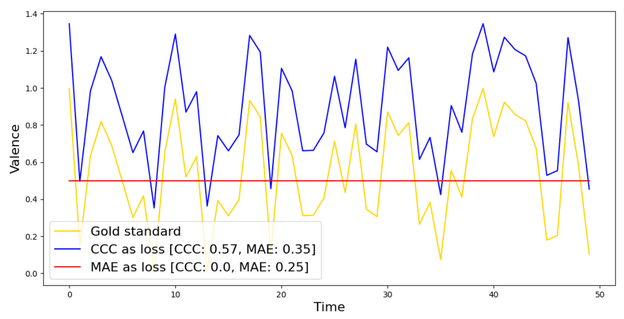

Figure 4: Example time series - the red plot performs better for $\mathcal{L}_{MAE}$ but worse for $\mathcal{L}_{CCC}$ than the blue.

seed	$\mathcal{L}_{MAE}$	seed	$\mathcal{L}_{MAE}$ + MCD	seed	$\mathcal{L}_{tilted}$
314	.2721	315	.4123	314	.4648
315	.2578	318	.4425	319	.4329
317	.4659	319	.4495	320	.4512
319	.3066	320	.4160	321	.4048
323	.396	322	.4608		
Mean	.3341		.4366		.4648
Std.	.0876		**.0343**		**.0424**

Table 3: Seeds with CCC test scores ≥ 0.1 for experiments based on $\mathcal{L}_{MAE}$

happens to the seeds predictions around zero in table 1. So, this large differences are not a sign of uncertainty, but they are caused by a not matching loss function, that does not take the time series characteristics of the labels into account.

To check anyway, if Monte Carlo Dropout or Quantile Regression were able to decrease uncertainty, we now just look at the seeds that did not tap into a local minimum, caused by the not suitable MAE-loss landscape and instead, learned the actual problem (in table 3 from table 1).

Without the erupting seeds, there is a measurable improvement of standard deviation with Monte Carlo Dropout and Quantile Regression towards raw $\mathcal{L}_{MSE}$.

Further, the seeds that do not predict around zero vary even if the architecture is exactly the same (with Dropout the same as without) or similar (for Quantile Regression only the last layer varies), because the convergence inside the optimization landscape also depends on small changes during training. This shows the instability of the seeds, respectively their convergences in the MAE loss landscape.

In summary, we must state that the choice of an appropriate loss function is essential to decrease uncertainty among seeds, because large varying local minimum in optimization space directly cause predictive uncertainty among seeds, which can be seen as the opposite of robustness.

4.2 Monte Carlo Dropout

As you can see in table 4, the standard deviation with Monte Carlo Dropout among seeds (trained with $\mathcal{L}_{CCC}$) is only half as much as without. And also, as lifted out in table 3, the standard deviation among seeds trained with $\mathcal{L}_{MAE}$, that did not drop towards zero, could be more than halved with Monte Carlo Dropout. So, MC-Dropout was able to significantly decrease predictive uncertainty.

We can observe in table 5, that the standard deviation

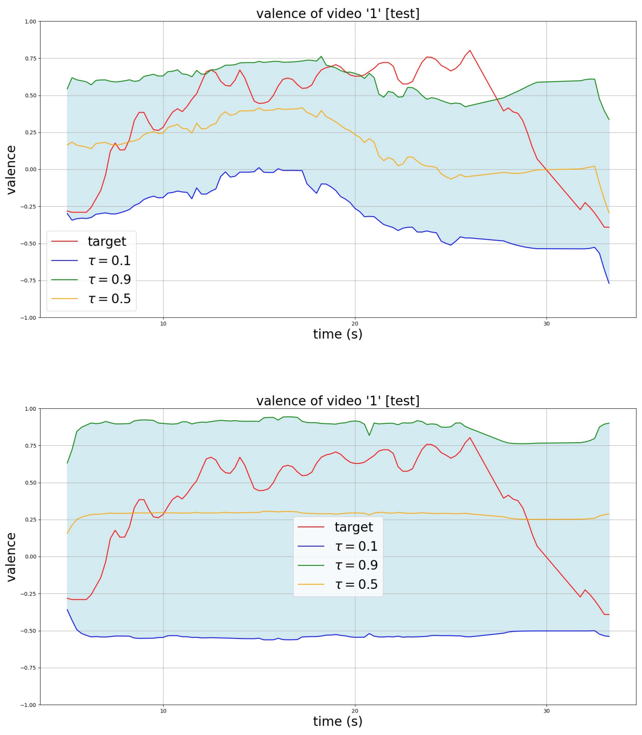

Figure 5: This figure shows the same sequence, predicted once by seed 314 (at the top) and the other time by seed 315 (underneath), both trained with $\mathcal{L}_{tilted}$. This visualizes on the one hand, how predictive uncertainty could be quantified with Quantile Regression, namely the gap between the upper and the lower quantile. And on the other hand, it clarifies the difference between a seed that dropped towards zero ($CCC_{315} = 0.0027$) and one that didn't ($CCC_{314} = 0.4648$), as can be seen in table 1 and indicated in theory in figure 4.

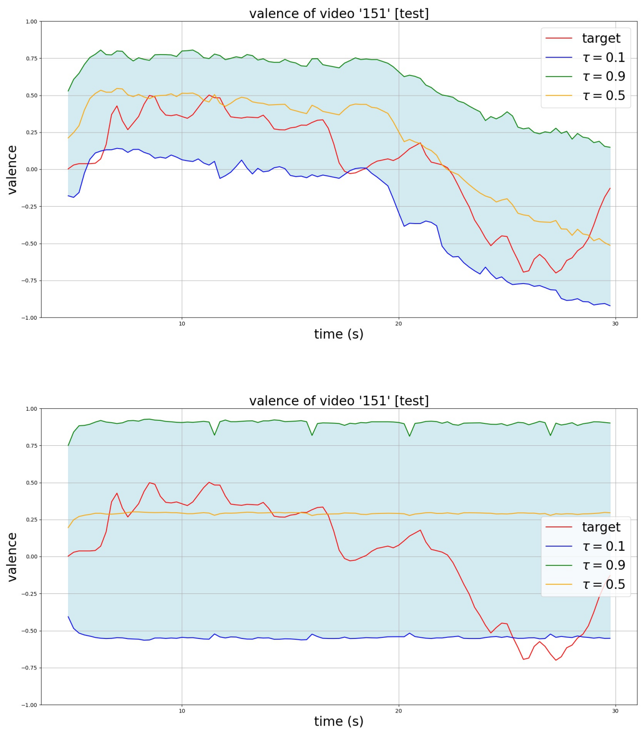

Figure 6: This figure shows the exact same situation as figure 5, but for another sample from the test set. In this, like each other sample predicted by seed 315 (stuck in local minimum), the mean is predicted, while seed 314 (not stuck in local minimum) tried to learn the real emotional state. Further, if you look closely at the above plot, you see that the quantified uncertainty by Quantile Regression for this seeds prediction (visualized by the light-blue area), is lower at the first half of time steps, where the predictions (the orange plot) are better, and is getting higher in the second half of the plot, where the quality of the predictions is getting worse. In our implementation of Quantile Regression the real prediction would be the mean of all plots, but $mean(green, orange, blue) \approx orange$.

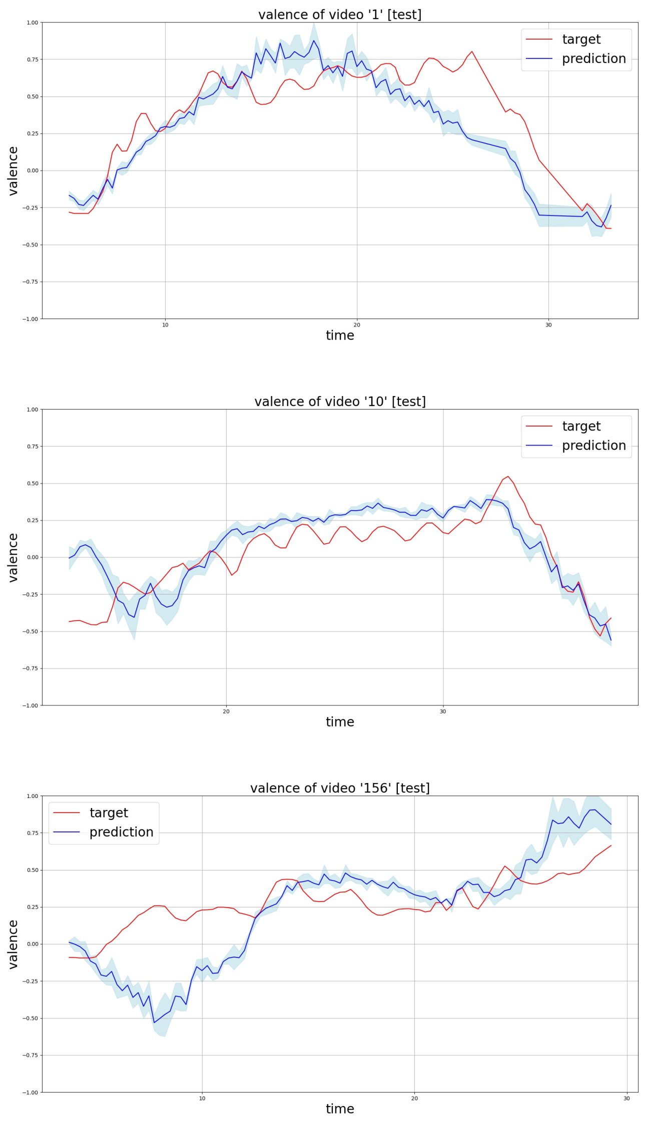

Figure 7: Monte Carlo Dropout sample predictions from experiment with $\mathcal{L}_{CCC} + MCD$ from table 4. - In the figures, the red plot is the label for samples from the test set. Blue represents the mean of the predictions from the model and light-blue is the standard deviation among predictions. The single forward run predictions (the light-blue area / standard deviation among predictions of one seed) might vary from seed to seed, but with MC-Dropout, the means (the final predictions) are moving closer together, as standard deviation among multiple seeds decreases.

	$\mathcal{L}_{CCC}$		$\mathcal{L}_{CCC}$ + MCD		$\mathcal{L}_{cwCCC}$	
seed	test	dev	test	dev	test	dev
314	.5852	.4582	.5685	.3996	.5549	.4260
315	.5673	.4463	.5569	.4290	.5596	.4020
316	.5525	.4294	.5618	.4361	.5694	.4279
317	.5459	.4215	.5574	.4213	.5572	.4285
318	.5576	.4262	.5682	.4377	.5586	.4221
319	.5618	.4398	.5677	.4309	.5619	.4236
320	.5444	.4270	.5525	.4124	.5500	.4321
321	.5757	.4426	.5723	.4417	.5464	.4250
322	.5659	.4086	.5633	.4149	.5456	.4123
323	.5501	.4164	.5566	.4022	.5278	.4225
Mean	.5606	.4316	.5625	.4226	.5531	.4222
Std.	.0132	.0150	**.0065**	.0149	**.0115**	.0088
Min.	.5444	.4086	.5525	.3996	.5278	.4020
Max.	.5852	.4582	.5723	.4417	.5694	.4321

Table 4: CCC scores for experiments with bert-4 based on $\mathcal{L}_{CCC}$

	$\mathcal{L}_{CCC}$		$\mathcal{L}_{CCC}$ + MCD		$\mathcal{L}_{cwCCC}$	
seed	test	dev	test	dev	test	dev
314	.3884	.4170	.3969	.4552	.4002	.4266
315	.3953	.4176	.3711	.4174	.3937	.4310
316	.3987	.4440	.3881	.4146	.3920	.4479
317	.3996	.4342	.3944	.4247	.3900	.4358
318	.4080	.4461	.3929	.4029	.3817	.4493
319	.3929	.4140	.3886	.4304	.4050	.4184
320	.4040	.4426	.3809	.4587	.3824	.4247
321	.3790	.4133	.3991	.4206	.3947	.4271
322	.3976	.4539	.3896	.4458	.4130	.4339
323	.4019	.4596	.3937	.4253	.4014	.4141
Mean	.3965	.4342	.3895	.4296	.3954	.4309
Std.	.0083	.0175	.0083	.0182	.0098	.0114
Min.	.3790	.4133	.3711	.4029	.3817	.4141
Max.	.4080	.4596	.3991	.4587	.4130	.4493

Table 5: RMSE scores for experiments with bert-4 based on $\mathcal{L}_{CCC}$

among seeds of their RMSE scores could not be decreased. But this phenomena is covered by the previous section 4.1, because just like MAE, the RMSE is not suitable to evaluate our *time-series-shaped* prediction, as both MAE and RMSE look at each time step separated and not as part of the whole (as explained in detail above in section 4.1).

4.3 Correlation-weighted CCC

As seen in table 4, reducing the loss for samples with large uncertainty among seeds leads to a small decreasing of the mean CCC score, but also a reduction of standard deviation (uncertainty among seeds) in contrast to the non-weighted $\mathcal{L}_{CCC}$.

The decreasing of the mean is comprehensible, because the model isn't forced to learn different samples elaborately (because of reduced loss), so it can't be that good at predicting similar examples than a model trained with the usual $\mathcal{L}_{CCC}$.

4.4 Quantile Regression

Quantile Regression was able to reduce predictive uncertainty (as seen in table 3), but it has to be evaluated in context of significantly different convergence directions of

	$\mathcal{L}_{CCC}$		$\mathcal{L}_{CCC}$ + MCD		$\mathcal{L}_{cwCCC}$	
seed	test	dev	test	dev	test	dev
314	.3168	.2322	.2705	.2025	.3007	.2258
315	.3225	.2192	.2920	.2006	.3254	.2244
316	.2994	.1937	.3219	.2265	.2962	.2116
317	.3254	.2079	.3293	.2141	.2964	.2319
318	.2991	.2428	.3269	.2307	.2673	.2274
319	.2852	.1989	.2841	.2086	.2928	.2362
320	.2757	.2036	.3168	.2172	.2801	.2270
321	.3503	.2230	.3303	.1976	.2906	.2178
322	.3302	.2409	.3228	.2283	.2894	.2241
323	.3108	.2450	.3125	.2178	.3272	.2362
Mean	.3115	.2207	.3107	.2144	.2966	.2262
Std.	.0223	.0191	**.0210**	.0119	**.0183**	.0076
Min.	.2757	.1937	.2705	.1976	.2673	.2116
Max.	.3503	.2450	.3303	.2307	.3272	.2362

Table 6: CCC scores for experiments with fastText based on $\mathcal{L}_{CCC}$

	$\mathcal{L}_{CCC}$		$\mathcal{L}_{CCC}$ + MCD		$\mathcal{L}_{cwCCC}$	
seed	test	dev	test	dev	test	dev
314	.4902	.5062	.4854	.5058	.4820	.4565
315	.4929	.5110	.4898	.5343	.4940	.5096
316	.4813	.4994	.4799	.4898	.4764	.4902
317	.4709	.5228	.4764	.4653	.4805	.4823
318	.4887	.4943	.4822	.5030	.4834	.4892
319	.4836	.5130	.4898	.4909	.4852	.4779
320	.4768	.4855	.5200	.5214	.4952	.4770
321	.4836	.5012	.4851	.4962	.4904	.5228
322	.4734	.4741	.4898	.4868	.4778	.4915
323	.4697	.5091	.4876	.4974	.4686	.4627
Mean	.4811	.5017	.4886	.4991	.4834	.4860
Std.	.0082	.0142	.0119	.0190	.0083	.0198
Min.	.4697	.4741	.4764	.4653	.4686	.4565
Max.	.4929	.5228	.5200	.5343	.4952	.5228

Table 7: RMSE scores for experiments with fastText based on $\mathcal{L}_{CCC}$

the MAE optimization landscape among seeds as described in detail in section 4.1.

4.5 bert-4 vs. fastText

As described above (section 3.1.2), bert-4 features are the output values of the last 4 layers from BERT fed with the transcriptions.

We expected, that the performance increases as the features are getting more complex and higher quality. Seeds trained on bert-4 features (table 4) perform much better than seeds trained on features generated with fastText (table 6).

And as the fastText features are of lower information content for the model, the uncertainty among seeds increases in contrast to bert-4.

Nevertheless, techniques, like Monte Carlo Dropout and re-weighting the loss for noisy samples, were (with bert-4 as well as with fastText) able to decrease predictive uncertainty.

Table 8 and 9 also confirm the claims from section 4.1. As the fastText features are simpler and consequently less meaningful, the seeds tend even more to converging towards predicting the mean.

	$\mathcal{L}_{MAE}$		$\mathcal{L}_{MAE}$ + MCD		$\mathcal{L}_{tilted}$	
seed	test	dev	test	dev	test	dev
314	.1305	.0997	.1707	.1150	.0003	.0023
315	.0022	.0041	.0792	.0788	.0833	.0811
316	.0413	.0558	.0081	.0266	.0003	.0008
317	.0009	.0112	.0006	.0108	.0007	.0023
318	.0031	.0042	.0463	.0549	.0004	.0008
319	.1387	.1138	.0593	.0961	.0005	.0024
320	.2251	.1498	.0538	.0390	.0005	.0011
321	.0327	.0370	.0816	.0602	.0003	.0010
322	.0020	.0039	.0010	.0079	.0007	.0014
323	.0010	.0017	.0006	.0017	.0004	.0032
Mean	.0578	.0481	.0501	.0491	.0087	.0096
Std.	.0792	.0546	.0533	.0389	.0262	.0251
Min.	.0009	.0017	.0006	.0017	.0003	.0008
Max.	.2251	.1498	.1707	.1150	.0833	.0811

Table 8: CCC scores for experiments with fastText based on $\mathcal{L}_{MAE}$

	$\mathcal{L}_{MAE}$		$\mathcal{L}_{MAE}$ + MCD		$\mathcal{L}_{tilted}$	
seed	test	dev	test	dev	test	dev
314	.4345	.4335	.4296	.4246	.4423	.4104
315	.4458	.4181	.4444	.4136	.4297	.4093
316	.4478	.4097	.4529	.4121	.4415	.4361
317	.4447	.4519	.4414	.4509	.4404	.4088
318	.4426	.4077	.4391	.4065	.4422	.4104
319	.4394	.4207	.4462	.4089	.4402	.4088
320	.4462	.4478	.4409	.4251	.4412	.4104
321	.4490	.4042	.4414	.4262	.4410	.4170
322	.4448	.4102	.4425	.4078	.4409	.4130
323	.4421	.4203	.4435	.4117	.4421	.4077
Mean	.4437	.4224	.4422	.4187	.4402	.4132
Std.	.0043	.0167	.0059	.0135	.0037	.0085
Min.	.4345	.4042	.4296	.4065	.4297	.4077
Max.	.4490	.4519	.4529	.4509	.4423	.4361

Table 9: RMSE scores for experiments with fastText based on $\mathcal{L}_{MAE}$

5. FUTURE WORK

Uncertainty is a widely studied topic and there are further approaches and techniques to avoid, reduce or measure uncertainty that were not implemented in this work. In the following sections we describe some of them that could be worked out in further studies.

5.1 Tilted CCC

We saw that Quantile Regression can reduce predictive uncertainty (table 3). As explained in section 1.1.1, quantile regression is based on the Mean-Absolute-Error, which doesn't pay attention to the time series characteristics of the labels. We propose to transfer the idea of tilting a fundamental loss function from $\mathcal{L}_{MAE}$ to other losses, like $\mathcal{L}_{CCC}$. So could the uncertainty-reducing effect of Quantile Regression be used without dispensing a suitable loss function for the given problem.

We did not evaluate the exact formula below yet, but as mentioned in section 2.1, the CCC ranges from -1 to $+1$, so at first glance the formula can be adapted from usual $\mathcal{L}_{tilted}$ easily:

$$\mathcal{L}_{tiltedCCC} = \frac{1}{N} * \sum_{i=1}^{N} max \\ (\tau_i * \mathcal{L}_{CCC_i}, (\tau_i - 1) * \mathcal{L}_{CCC_i}), \tag{11}$$

in which τ_i denotes the i-th quantile and $\mathcal{L}_{CCC_i}$ the $\mathcal{L}_{CCC}$ computed for the i-th output node.

But probably this approach is too naive, because at CCC score -1 indicates a strong negative correlation and $+1$ a strong (positive) correlation, with it, in real training runs, positive values will appear much more often. After some time the models (hopefully) knows how to predict a not-negative-correlating output series and just learns how to increase the correlation. This would mean that the quantiles have no effect and the loss "mutates" to ordinarily $\mathcal{L}_{CCC}$ (for each output node). This would maybe only affect training at the very beginning.

So we could think to shift the quantiles. Assumed that after some time of successful learning the $\mathcal{L}_{CCC}$ is roughly limited from 0 to $+1$, we could shift the range of values of $\mathcal{L}_{CCC}$ by:

$$\mathcal{L}_{tiltedCCC} = \frac{1}{N} * \sum_{i=1}^{N} max \\ (\tau_i * (\mathcal{L}_{CCC_i} - 0.5), (\tau_i - 1) * (\mathcal{L}_{CCC_i} - 0.5)). \tag{12}$$

It should be noted that the added factor 0.5 might not be the best. The choice of if determines the quantiles and has to be chosen wisely and in dependency of the under-laying fundamental loss function (in our case $\mathcal{L}_{CCC}$).

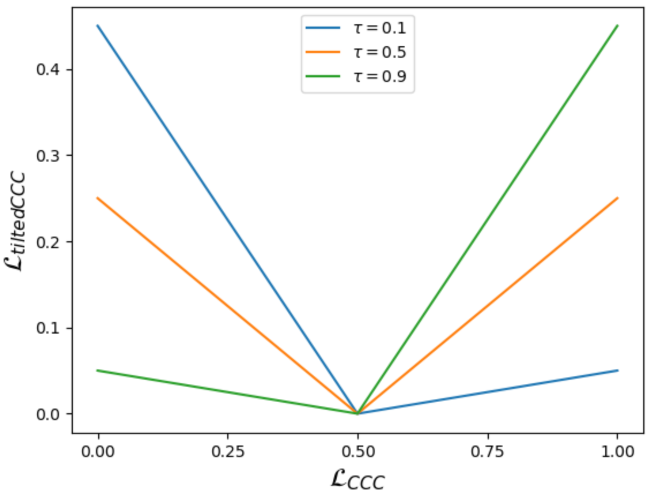

Figure 8: Transferring the tilted character to $\mathcal{L}_{CCC}$.

The formula with a variable factor-shift λ looks as follows:

$$\mathcal{L}_{tiltedCCC} = \frac{1}{N} * \sum_{i=1}^{N} max \\ (\tau_i * (\mathcal{L}_{CCC_i} - \lambda), (\tau_i - 1) * (\mathcal{L}_{CCC_i} - \lambda)), \tag{13}$$

with λ ranging from 0 to $+2$ (in terms of $\mathcal{L}_{CCC}$)[2] and used to determine the stressed quantiles.

As the performance of the model increases, one could think about adjusting λ during training time.

5.2 More features and configurations

We have seen that there are large differences between fastText and bert-4. Both are features extracted from transcription, so written text, and in further works it could be illuminating to repeat some experiments with features extracted from audio (e.g. VGGish [10]) or visual features or to feed multiple of those features to the model.

Also, we could vary hyperparameter used in techniques to deal with predictive uncertainty. For example, we could test if the predicted measurable uncertainty by Monte Carlo Dropout increases with higher dropout rates or if varying quantiles lead to varying uncertainty by Quantile Regression. The previous experiments only scratch the surface of the feasible with these techniques.

5.3 Quantifying predictive uncertainty

The used methods were applied to reduce predictive uncertainty among seeds, respectively make the model more robust. We did not measure and return the uncertainty of a given inference run, although there might be a lot of applications for that. An application that uses the neural network to make predictions might need to know the certainty for the predicted output. Further, the output could be discarded, if the measured uncertainty is too high. With other words: the neural network should be able to say "I don't know"[3].

[2] If CCC score is -1 (highly negative correlated) the $\mathcal{L}_{CCC}$ becomes $1 - (-1) = 2$.

[3] See epistemic uncertainty as mentioned in section 1

Luckily, the MuSe-CaR dataset contains multiple annotations and if we wanted we could train a neural network to directly map a sample to the uncertainty among the annotations (e.g. represented by their standard deviation) as described in the section 1.1.2 about distributional parameter estimation. But if the data doesn't include multiple annotations we can not apply such techniques.

Therefore, we could think about ways to quantify the uncertainty elsewhere. For example, during inference with Monte Carlo Dropout, we could return the mean as final prediction as before and additionally calculate the standard deviation among the individual forward runs for representing the model's uncertainty, as visualized in figure 7.

6. CONCLUSIONS

We saw multiple techniques to decrease predictive uncertainty during a complex emotion modelling process. The three main techniques evaluated within this work are Monte Carlo Dropout, Quantile Regression and re-weighting the loss for noisy labels.

Emotions, especially continuous emotional states, aren't easy to predict, hence different seeds show some unexpected results. The differences among seeds can be huge, which suggests that small changes on the model or configuration could also have large impact on the models performance and predictive uncertainty. The most outstanding observation are the seeds dropping towards zero when $\mathcal{L}_{MAE}$ was used as loss criterion, because this clarifies the importance of a loss function that really maps the problem in a (similar) way the performance metrics will do during evaluation. We can conclude that an appropriate modelling of the predicted problem is more important for reduction of uncertainty than any technique to do so. But we also saw that such techniques can basically decrease predictive uncertainty. Furthermore, we nudged to transfer the idea of a very common uncertainty-approach, namely Quantile Regression, to other loss functions, so it could be used respecting problem specific label characteristics (that MAE can't always cover). The observations from table 3 suggest that Quantile Regression has great potential in reducing predictive uncertainty. So moving its characteristics to problem specific optimization landscapes might bring advantage in decreasing predictive uncertainty.

7. ACKNOWLEDGMENTS

Thanks to Lukas Stappen for originating the project and weekly feedback, as well as Lea Schumann for explanations about the dataset and providing baseline code for prediction of the emotional state, which was extended with techniques to deal with predictive uncertainty.

References

[1] Luis Aguado et al. "Learning about Faces: Effects of Trustworthiness on Affective Evaluation". In: *The Spanish journal of psychology* (2011).

[2] Atsushi Ando et al. "Soft-Target Training with Ambiguous Emotional Utterances for DNN-Based Speech Emotion Classification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2018).

[3] Simon Bachstein. *Uncertainty Quantification in Deep Learning.* 2019. URL: `https://sbachstein.de/master_thesis.pdf`.

[4] Charles Blundell et al. "Weight Uncertainty in Neural Networks". In: Proceedings of Machine Learning Research (2015).

[5] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* (2017).

[6] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019).

[7] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of The 33rd International Conference on Machine Learning* 48 (2016).

[8] Michael Grimm and Kristian Kroschel. "Evaluation of natural emotions using self assessment manikins". In: *Proceedings of ASRU* (2005).

[9] Jing Han et al. "Exploring Perception Uncertainty for Emotion Recognition in Dyadic Conversation and Music Listening". In: *Cognitive Computation* (2020).

[10] Shawn Hershey et al. "CNN Architectures for Large-Scale Audio Classification". In: *International Conference on Acoustics, Speech and Signal Processing* (2017).

[11] Phuc Le-Khac, Graham Healy, and Alan Smeaton. "Contrastive Representation Learning: A Framework and Review". In: *IEEE Access* 8 (2020).

[12] Roger Koenker. *Quantile Regression.* Econometric Society Monographs. 2005.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017).

[14] Vikram Mullachery, Aniruddh Khera, and Amir Husain. *Bayesian Neural Networks.* 2018. arXiv: `1801.07710`.

[15] Nagarajan Natarajan et al. "Cost-Sensitive Learning with Noisy Labels". In: *Journal of Machine Learning Research* (2018).

[16] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. *Confident Learning: Estimating Uncertainty in Dataset Labels.* 2020. arXiv: `1911.00068`.

[17] James Russell. "A Circumplex Model of Affect". In: *Journal of Personality and Social Psychology* (1980).

[18] Lukas Stappen et al. "MuSe 2020 Challenge and Workshop: Multimodal Sentiment Analysis, Emotion-Target Engagement and Trustworthiness Detection in Real-Life Media: Emotional Car Reviews in-the-Wild". In: *Proceedings of the 1st International on Multimodal Sentiment Analysis in Real-Life Media Challenge and Workshop* (2020).

[19] Lukas Stappen et al. "Uncertainty-Aware Machine Support for Paper Reviewing on the Interspeech 2019 Submission Corpus". In: *Proceedings Interspeech* (2020).

[20] Licai Sun et al. “Multi-Modal Continuous Dimensional Emotion Recognition Using Recurrent Neural Network and Self-Attention Mechanism”. In: *Proceedings of the 1st International on Multimodal Sentiment Analysis in Real-Life Media Challenge and Workshop* (2020).

[21] Michel Valstar et al. “AVEC 2016 - Depression, Mood, and Emotion Recognition Workshop and Challenge”. In: *Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge* (2016).

[22] Mike Wu and Noah Goodman. *A Simple Framework for Uncertainty in Contrastive Learning.* 2020. arXiv: `2010.02038`.

[23] Cihang Xie et al. *Smooth Adversarial Training.* 2020. arXiv: `2006.14536`.

[24] Yi-Hsuan Yang and Homer H. Chen. “Machine Recognition of Music Emotion: A Review”. In: *ACM Transactions on Intelligent Systems and Technology* (2012).