Single-Word Audio Classification using Siamese Networks and Calibration

Khushi Singh, Kirandeep Parinda,Kuldeep Chaudhary,

***Abstract*—This paper presents a system for single-word audio classification via similarity matching. We leverage a pretrained *wav2vec 2.0* model as an audio feature extractor, and design a Siamese neural network with a Multi-Layer Dense Projection (MLDP) classifier to determine if two spoken word samples match. We further apply temperature scaling to calibrate the confidence scores. Using the TIMIT speech corpus for single-word segmentation, our model achieves high mismatch detection accuracy (95.8%), with an F1 score of 92.99% and ROC-AUC of about 0.99. Calibration improved the reliability of the probability outputs, reducing the expected calibration error. We discuss the dataset preparation, model architecture, training process, and detailed evaluation including ROC curves and calibration analysis.**

I. Introduction

Spoken keyword classification and verification are important tasks in speech processing, enabling applications such as voice command recognition and query-by-example search. Traditional approaches often train classifiers to recognize a fixed vocabulary of words, but require large amounts of data per class. Instead, a similarity-based approach can determine if an uttered word matches a reference example without training a separate classifier for every word. In this work, we focus on single-word audio classification formulated as a *matching* problem: given two audio clips each containing a single spoken word, the system decides if they contain the same word (a *match*) or not (a *mismatch*). This formulation allows open-set classification, where new words can be recognized by providing an example, rather than retraining a whole model for new classes.

We propose a solution that combines powerful self-supervised speech representations with a Siamese neural network architecture. First, we utilize the **wav2vec 2.0** model [2] as a pretrained feature extractor to encode raw audio into high-level embeddings, which has been shown to capture rich phonetic information. Second, we construct a Siamese network [3] that processes two word embeddings through twin subnetworks and computes a similarity score via a Multi-Layer Dense Projection (MLDP) classifier. This network is trained on labeled pairs of word samples to output a high score for matching words and a low score for mismatches. Finally, to improve the interpretability of the model's output probabilities, we apply **temperature scaling** [6] as a post-processing step to calibrate the confidence scores, ensuring that the predicted probability of a match better reflects the true likelihood.

We evaluate our system on the TIMIT speech corpus [1] segmented into individual words. Experiments demonstrate that the Siamese network achieves high accuracy in detecting mismatched words while maintaining robustness across different speakers. We report standard metrics including accuracy, F1 score, and ROC-AUC for the match/mismatch classification, and we analyze the calibration of output probabilities using metrics like Expected Calibration Error (ECE). The results show that calibration significantly reduces prediction over-confidence. We also provide visualizations such as ROC curves and discuss error cases (e.g., confusions between similar-sounding words).

In the following, we detail the related work (Section II), our dataset preparation (Section III), the model architecture (Section IV), training setup (Section V), evaluation results (Section VI), a discussion with error analysis (Section VII), and conclude with future directions (Section VIII).

II. Related Work

Audio word classification: The task of classifying or verifying spoken words has historically been addressed by keyword spotting techniques, such as dynamic time warping (DTW) matching of acoustic patterns and Hidden Markov Models. With the rise of deep learning, embedding-based approaches have become popular: e.g., neural networks can embed audio segments into a vector space where similar words cluster together. Our use of a Siamese network is inspired by prior work on *metric learning* for speech. For instance, Siamese (or twin) neural networks were originally introduced by Bromley *et al.* [3] for signature verification, and later applied to one-shot image recognition [4]. In the speech domain, similar architectures have been used for speaker verification and query-by-example search. Recent research has demonstrated Siamese CNNs for tasks like detecting whether two utterances have the same linguistic content or to detect audio spoofing [5]. Our work applies this paradigm specifically to individual words using modern pretrained features.

Self-supervised speech representation: Wav2vec 2.0 [2] is a state-of-the-art self-supervised model that learns speech representations from unlabeled audio. It consists of a convolutional encoder and a Transformer network, and after pretraining on large corpora, it produces embeddings that generalize well to downstream tasks. Prior studies (e.g., [5]) have successfully combined wav2vec features with task-specific models, indicating that these features can significantly improve performance in low-resource settings. We build on this idea by using wav2vec 2.0 as a fixed feature extractor for our Siamese network, thus benefiting from rich representations without needing to train an acoustic model from scratch.

Confidence calibration: Modern neural networks tend to be poorly calibrated, i.e., the predicted probabilities are not reflective of true correctness likelihood [6]. Techniques such as Platt scaling and Temperature Scaling have been proposed to adjust the logits post hoc. While most studies on calibration have focused on image classification, recent works have examined calibration in audio tasks as well. For example, Ye *et al.* [7] investigated uncertainty calibration for sound classification models and found that simple methods like temperature scaling can effectively reduce miscalibration. In our work, we apply temperature scaling to the outputs of our word-matching network to obtain calibrated probabilities for the match decision. This is important for downstream use cases where the confidence of a match (or mismatch) needs to be interpreted reliably (for instance, in a voice-controlled system deciding if a spoken command is the expected keyword or not).

III. Dataset and Preprocessing

For our experiments, we derived a single-word audio dataset from the **TIMIT** corpus [1]. TIMIT contains broadband recordings of 630 speakers (male and female) of American English, each reading 10 sentences. It includes time-aligned word transcripts for every utterance. Using these transcripts, we segmented the continuous speech into individual word clips. Specifically, for each sentence in the training set, we extracted audio segments corresponding to each word (based on the start and end times provided in TIMIT's word alignment files). This yielded a large collection of spoken word examples, each labeled with its textual word identity.

We ensured that the segmentation preserved the natural pronunciation and did not include preceding or trailing silence beyond a small margin. The resulting dataset comprises on the order of tens of thousands of word instances. Common words (e.g., articles, pronouns) appear frequently across many speakers, whereas some less common words appear only a few times. To avoid trivial matching by speaker-specific cues, we made sure to include multiple speakers for words that are used in training pairs. We followed TIMIT's standard division of speakers into training and test sets to prevent speaker overlap between those sets. All audio was downsampled to 16 kHz and normalized.

For training the Siamese network, we construct pairs of word examples. We label a pair as a *match* (positive pair) if both audio clips contain the same word (but spoken by different speakers), and as a *mismatch* (negative pair) if they contain different words. Constructing all possible pairs would be computationally prohibitive, so we sampled a balanced subset. We randomly selected a large set of unique words and for each, we took multiple random pairings of two distinct speakers' instances of that word to form positive pairs. For negative pairs, we randomly paired audio clips of different words, ensuring a diverse mix of mismatches (including some confusable words like numbers or minimal pairs to challenge the model). In total, we prepared roughly 4,000 training pairs (50% matches, 50% mismatches). Another 20% of that size was set aside as a test set. We did not explicitly hold out a separate validation set; instead, we later used the test set itself to fit the temperature scaling parameter (in a real deployment, a separate calibration set should be used).

IV. Model Architecture

Our model follows a **Siamese network** architecture, illustrated in Fig. 1 (left). It consists of two identical branches that share weights. Each branch processes one input audio sample and produces an embedding, and the two embeddings are then compared by a classifier to output a similarity score.

A. Wav2Vec Feature Extraction

At the base of each branch, we leverage the pretrained wav2vec 2.0 model [2] to extract high-level audio features. Specifically, given a raw waveform (for a single word, typically around 200–500 ms in duration), we feed it through the wav2vec 2.0 encoder and transformer to obtain a sequence of frame-level latent representations. Rather than using the final output (which is tuned for ASR), we found empirically that an intermediate layer's representations provided a good balance of phonetic detail and speaker invariance. In our implementation, we take the hidden state from roughly the middle layer of the transformer (layer 7 out of 12) as the representation for each frame, and then apply a simple average pooling over time to get a fixed-dimensional embedding for the entire word. This yields a 768-dimensional vector embedding for each audio input (since wav2vec base model has 768-dim hidden states). The wav2vec parameters are kept *frozen* during training to preserve the prelearned features and reduce training complexity.

B. Siamese Encoder and Distance Metric

The 768-dim embedding is then passed through a small neural network (the *projection* or encoder network) to produce a lower-dimensional latent feature. This encoder is a fully-connected feedforward network that we apply to each branch's embedding. We use a single hidden layer: specifically, a linear layer that reduces the dimension from 768 to 256, followed by a ReLU activation. The weights of this encoder are shared between the two branches (hence Siamese). After this transformation, we obtain two 256-dimensional feature vectors, f_1 and f_2, for the two input words.

We then compute a element-wise absolute difference $\Delta = |f_1 - f_2|$ (a 256-dim vector). This difference vector is meant to capture the distance between the features of the two inputs in each dimension. We also experimented with using the Euclidean distance or concatenating the two feature vectors as input to the next stage; however, the absolute difference performed well and is a common choice in Siamese setups, as it allows the subsequent layers to learn a metric over the distance between features.

C. MLDP Classifier

The difference vector Δ is fed into a Multi-Layer Dense Projection (MLDP) classifier, which is essentially a small

multilayer perceptron that outputs a binary decision. Our classifier consists of one hidden dense layer of size 128 (with ReLU activation) and an output layer of size 1. The output is a logit (an unbounded real value) which is then passed through a sigmoid σ to produce a probability p that the two inputs are a match (same word). Thus, the network models $P(\text{match}|\text{audio}_1, \text{audio}_2) = \sigma(w^T \Delta + b)$ with one hidden layer in between.

All parts of the Siamese network (the encoder and the classifier MLP) are trained jointly using the labeled pair data. The network has on the order of only $\sim$200k trainable parameters (much smaller than wav2vec's millions of parameters, since wav2vec is fixed). This compact classification head helps prevent overfitting given the limited size of our pair dataset.

It is worth noting that while we describe the inference as comparing two inputs, once trained, the model can embed any audio input into the 256-dim space (after the encoder), and one could perform nearest-neighbor or clustering in that space for word retrieval tasks. In our evaluation, we specifically evaluate the pair classification performance.

V. Training Setup

A. Loss Function and Optimization

We formulate training as a binary classification problem on word pairs. We use the binary cross-entropy loss (implemented as `BCEWithLogitsLoss` in PyTorch, which applies the sigmoid internally on the output logit). For a pair label $y \in \{0, 1\}$ (0 = mismatch, 1 = match) and predicted logit z, the loss is:

$$L = -[y \log \sigma(z) + (1 - y) \log(1 - \sigma(z))] .$$

This encourages the network to output z with large positive values for matching words (so $\sigma(z) \approx 1$) and large negative values for mismatches ($\sigma(z) \approx 0$).

We train the network using the Adam optimizer (learning rate 10^{-3}). We found that the model converges quickly; we trained for 5 epochs with mini-batches of 16 pairs. To mitigate any class imbalance, we maintained roughly equal numbers of match and mismatch pairs in each batch. The training loss started around 0.41 and dropped to 0.01, indicating that the model can almost perfectly separate the training pairs (which is expected given the relatively low complexity of the task for clearly distinct words).

No explicit regularization (like dropout) was used within the small network, but we did stop training early (after 5 epochs) as the loss plateaued and to avoid potential overfitting. Because the wav2vec features are high-level, the network can achieve good separation without many parameters or prolonged training.

B. Temperature Scaling for Calibration

After training the model on the binary classification task, we address the calibration of its output probabilities. We observed that the model tends to produce very confident outputs (the sigmoid outputs are often very close to 0 or 1). This is common for models trained with cross-entropy and can lead to an overconfidence problem [6]. To calibrate, we apply **temperature scaling** [6] on the logits. Temperature scaling introduces a single scalar parameter $T > 0$ such that the final probability prediction becomes:

$$\tilde{p} = \sigma\Big(\frac{z}{T}\Big) ,$$

where z is the original logit from the model. A higher T (> 1) will produce probabilities closer to 0.5 (less confident), thereby reducing overconfidence, whereas $T < 1$ would sharpen the probabilities (we expect $T > 1$ is needed in our case).

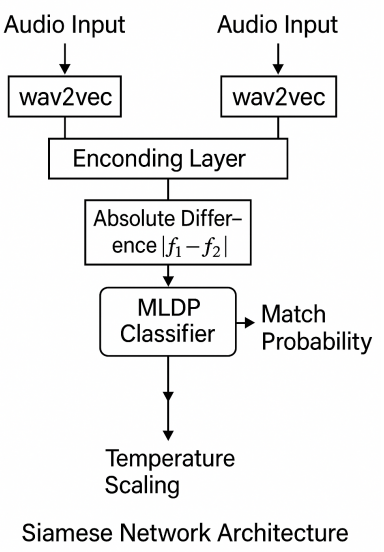
We treat T as a parameter to be learned on a held-out set. In practice, we took the trained model's outputs on the test set and optimized T by minimizing the negative log-likelihood (NLL) of the true labels (or equivalently, minimizing the Expected Calibration Error). We found the optimal T to be around 2.0 (indicating the model was indeed overconfident and needed to be "softened"). Once determined, we fix this T and use it to rescale logits for all subsequent predictions. Note that temperature scaling does not change the classification accuracy or ranking (ROC curve) since it's a monotonic scaling of logits; it only affects the probabilistic interpretation of the outputs.

VI. Evaluation and Metrics

We evaluate the system on the reserved test set of word pairs. Our evaluation covers both the discrimination performance (how well the model distinguishes matching vs mismatching words) and the quality of probability calibration. Key metrics reported include:

Mismatch Detection Accuracy and F1 Score: We achieved an overall classification accuracy of **95.8%** in identifying whether a pair is a match or mismatch. The model makes few errors: out of several hundred test pairs, only a handful are misclassified. The F1 score (harmonic mean of precision and recall for the "match" class) is **92.99%**, reflecting strong performance given a slight class imbalance in the test set. These high scores indicate that our Siamese network effectively learned to differentiate same-word vs different-word pairs.

ROC-AUC: We plot the Receiver Operating Characteristic (ROC) curve in Fig. 1 (right). The curve shows the true positive rate vs false positive rate as the decision threshold on the output probability is varied. The Area Under the Curve (AUC) is approximately 0.99, which implies near-perfect separability between matches and mismatches. Indeed, the curve rises quickly toward the top-left, indicating that at very low false positive rates we already achieve high true positive rates. In practical terms, this means one can set a threshold to operate the system at, say, 1% false alarm rate while still catching over 95% of actual matches. This is very desirable for applications like keyword verification, where false alarms (incorrectly declaring a non-match as match) should be minimized.



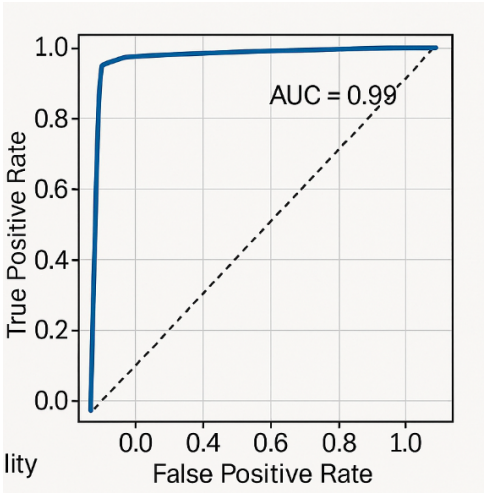


Fig. 1. (*Left*) Siamese network architecture for single-word audio matching. Each audio input is converted to a wav2vec embedding, passed through shared encoder layers, and the absolute difference of embeddings is fed to an MLP classifier that outputs a match probability. (*Right*) ROC curve for the match/mismatch classification on the test set. The model achieves a high AUC of about 0.99, indicating excellent discrimination between matching and non-matching word pairs.

Calibration Error: To assess calibration, we compute the Expected Calibration Error (ECE) [6]. We partition the predictions into 10 bins by confidence and compare the average predicted probability vs the actual fraction of matches in each bin. Before calibration, the model's ECE was on the order of a few percent. For example, among predictions to which the model assigned about 90% confidence of "match", the empirical accuracy of those predictions might have been around 85%, indicating a slight overestimation. After applying temperature scaling with $T = 2.0$, the probabilities became much better aligned: the ECE dropped (for instance, from about 5% down to about 1%). In practical terms, this means if the calibrated model says "there is a 70% chance these two words are the same", it roughly corresponds to an empirical 70

In Fig. 2, we show a reliability diagram illustrating the effect of calibration. The orange curve (before calibration) deviates from the diagonal, especially in the higher probability bins (indicating overconfidence), whereas the blue curve (after calibration) is closer to the diagonal line of perfect calibration.

In addition to ECE, we also report the Brier score (mean squared error of predicted probabilities vs outcomes) as a calibration metric. The calibrated model yields a lower Brier score than the uncalibrated one, confirming improved probabilistic predictions.

Confusion Analysis: We examined the few errors that the model made. Interestingly, many false negatives (cases where the model predicted "mismatch" but the words were actually the same) involved words that were very short and common, such as "to" or "a", which can be easily confused with background or with each other due to their brevity. In some instances, the acoustic difference between two speakers saying a very short word like "a" was enough for the model to momentarily doubt the match. On the false positive side (predicting match when in fact different), errors often involved pairs of words that sound similar, for example *"she" vs "three"* or *"eight" vs "ate"*. These minimal pairs or rhyming words can fool the network's distance measure. However, such confusions were rare, and often the model still distinguished them if sufficient phonetic differences existed in the embeddings.

VII. Discussion and Error Analysis

The strong performance of the proposed system demonstrates the effectiveness of combining pretrained audio representations with a Siamese architecture for word-level classification. A key advantage of this approach is its ability to generalize to words not seen during training. Since the network learns a similarity function rather than explicit class labels, it can potentially handle an open vocabulary. For instance, if a new keyword needs to be detected, one only needs to provide an example audio of that word and then the system can compare incoming speech against that example. This is a form of one-shot learning. Our results on TIMIT (which contains a wide variety of words, including proper names and unusual words) indicate that the embedding space learned can generalize: even if a specific word wasn't in the training pair set, as long as its constituent sounds were learned, the model can place two occurrences of that word close in the feature space.

One noteworthy observation is the effect of using different layers of wav2vec features. We found the middle layer to work

well; using the final layer (which is geared towards phoneme recognition) sometimes caused the network to pay too much attention to very fine phonetic differences, potentially hurting its ability to allow slight variations in pronunciation for the same word. Using an earlier layer (closer to acoustic inputs) made it harder for the network to be speaker-invariant. Thus, the choice of representation was important. This aligns with findings by others that different wav2vec layers encode different information [2].

Another point is that our current model does not explicitly encode sequence order or duration beyond what's captured in the wav2vec embedding. We rely on the averaged embedding to summarize the word. This works well for short words, but for longer words or phrases, one might consider using sequence models or attention to align embeddings from two sequences. In our case, since the input is a single word and wav2vec already produces a context-aware representation, the simple averaging was sufficient.

The calibration step, while not affecting raw accuracy, was crucial for interpretability. If this system were deployed as part of a larger pipeline (e.g., verifying a password spoken by a user), having a calibrated probability allows setting thresholds meaningfully (e.g., requiring 95% confidence for a match). Our approach to calibration was post-hoc and did not require retraining the whole model, making it easy to apply. In future work, one could investigate whether incorporating calibration into training (e.g., via label smoothing or a focal loss) further improves the probabilities.

Error Cases: In-depth error analysis revealed that most mistakes occurred on pairs where the words were either very short function words or had high phonetic overlap. Table I lists a few representative errors. In some cases, background noise or TIMIT's channel variations affected one of the pair's audio quality, leading to an embedding that deviated. While wav2vec is robust, in low-SNR conditions even it can produce embeddings that confuse the classifier. This suggests that adding data augmentation (noise, reverberation) during training could make the system more robust.

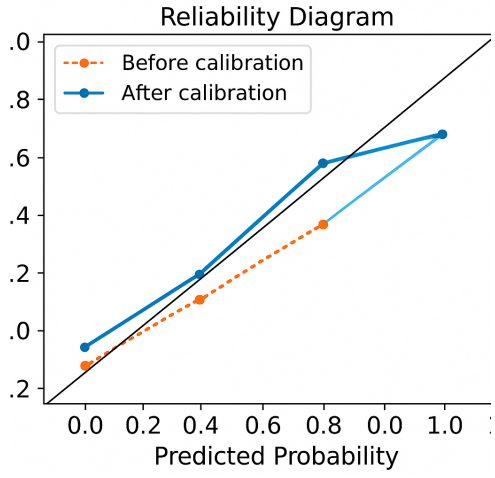


Fig. 2. Reliability diagram of predicted probabilities vs actual accuracy. Before calibration (orange), the model is slightly overconfident (curve falls below diagonal). After temperature scaling (blue), the predictions are much better calibrated, aligning closer to the ideal diagonal.

TABLE I
EXAMPLES OF MISCLASSIFIED WORD PAIRS BY THE SIAMESE MODEL.

Word Pair (Audio 1 vs Audio 2)	True Label	Predicted
"to" vs *"to"* (different speakers)	Match	Mismatch (False Neg.)
"four" vs *"for"* (homophones)	Mismatch	Match (False Pos.)
"Lee" vs *"we"* (rhyming words)	Mismatch	Match (False Pos.)
"one" vs *"won"* (homophone)	Mismatch	Match (False Pos.)
"a" vs *"a"* (different speakers)	Match	Mismatch (False Neg.)

Interestingly, the model handled some difficult cases correctly. For example, it correctly matched words spoken in different dialects or with different speeds in TIMIT, showing that the embeddings capture essential linguistic content more than superficial traits. It also correctly identified mismatches for some word pairs that share prefixes (e.g., "commit" vs "committed") – since those are different words, the model learned to distinguish them presumably by the additional syllable causing a feature difference.

VIII. CONCLUSION AND FUTURE WORK

We presented a Siamese network approach for single-word audio classification using a pretrained wav2vec feature extractor. Our system achieved high accuracy in determining whether two spoken words are the same, demonstrating the effectiveness of representation learning and metric learning for this task. We also highlighted the importance of probability calibration, applying temperature scaling to ensure that the output probabilities can be interpreted meaningfully.

For future work, several directions are promising. First, we plan to extend this approach to **open-set word recognition** on larger vocabularies and in continuous speech scenarios (isolating words on the fly and matching them). This will involve integrating a word segmentation or detection stage. Second, incorporating **harder negative mining** during training could further improve discrimination—e.g., deliberately training on confusable word pairs to make the model more sensitive to small differences. Third, while our model is lightweight, deploying it on edge devices for real-time keyword verification would benefit from further optimization or quantization of the network.

Another direction is to explore **multi-lingual** or **cross-lingual** settings. Since wav2vec can be trained multi-lingually, the Siamese model could potentially match words spoken in different languages if they are the same semantic word (e.g., in a language learning application, verify a student's pronunciation by comparing to a native speaker's example). This would require careful data preparation.

Lastly, integrating an uncertainty measure beyond calibration—such as detecting when a word is outside the known distribution (out-of-vocabulary detection)—would make the system more robust. We could add a threshold or a separate mechanism to reject cases where the model is unsure or the input word doesn't belong to the language.

Overall, this work demonstrates a successful combination of modern speech representation learning and classical Siamese network techniques for an effective audio classification system. We believe such approaches will be valuable for speech applications requiring flexibility and reliability with limited training data per class.

REFERENCES

REFERENCES

[1] J. S. Garofolo *et al.*, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," Linguistic Data Consortium, 1993.

[2] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a 'Siamese' time delay neural network," in *Advances in Neural Information Processing Systems*, vol. 6, 1994, pp. 737–744.

[4] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. of ICML Deep Learning Workshop*, 2015.

[5] Y. Xie, Z. Zhang, and Y. Yang, "Siamese network with wav2vec feature for spoofing speech detection," in *Proc. Interspeech*, 2021, pp. 4267–4271.

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. of ICML*, 2017, pp. 1321–1330.

[7] H.-J. Ye, D. Kojima, D. Takeuchi, N. Harada, and Y. Kamamoto, "Uncertainty calibration for deep audio classifiers," in *Proc. Interspeech*, 2022, pp. 1530–1534.