Development of Fish Variant Multi Object Detection System for Video based on YOLOv7

Muhammad Daffa Ajiputra1, Prima Dewi Purnamasari2  
Computer EngineeringUniversitas Indonesia  
mohammad.daffa99@ui.ac.id1, prima.dp@ui.ac.id2

*Abstract*— Indonesia is one of the world's largest exporters of fish, which exposes Indonesia's fishing sector to many threats. Illegal, unreported, unregulated (IUU) fishing is one of the problems that resulted in a significant impact in a form of a big loss that is created for the Indonesian fisheries sector. To prevent that problem, there are a lot of solutions that have been proposed, one of which is the application of technology such as surveillance cameras, but it still doesn't have a big impact to reduce and eliminate IUU fishing. Therefore, this research is conducted to develop a multi-object detection system for the detection of fish species based on YOLOv7, an artificial intelligence model that can detect a fish to supervise the number of fish that is caught by the fisherman so IUU fishing can reduce significantly. From the testing, the YOLOv7 model becomes the best YOLOv7 model variant that can be used to detect a fish with the value of mAP that can reach up to 86.1% and the value of inference time up to 14.5 ms that can produce an FPS total up to 69 FPS. The value can be achieved by doing some modifications in data annotation, the training model method, image size, and iteration on training. However, the YOLOv7 model has a very slow inference time up to 797.6 ms when it’s installed in Jetson Nano even though the detection accuracy has the same value.

Key words: deep learning, computer vision, object detection, YOLOv7

# Introduction

Indonesia is one of the world's largest fish exporting countries. With a value of 558 million US dollars [1], Indonesia is rated as the world's seventh largest fish exporting country for filleted and frozen fish in 2021. Tuna is one of Indonesia's high-value fish export commodities. The value of Indonesian tuna exports in 2021 was 325.4 million US dollars, with an export volume of 48.3 Mega Tons for filleted and frozen tuna [2].

On the other hand, due to the high number of Indonesian marine export commodities, the country is not immune to occurrences of illegal, unreported, and unregulated (IUU) fishing. In 2021, the Ministry of Maritime Affairs and Fisheries apprehended 167 Indonesian and foreign ships, including ships from Vietnam and Malaysia, for IUU fishing [3]. It is estimated that illegal, unreported, and unregulated fishing cost Indonesia 74 million US dollars in 2021 [4]. The Ministry of Maritime Affairs and Fisheries has developed multiple strategies to prevent IUU fishing on Indonesian territory. For example, eight Malaysian ships were sunk owing to unlawful fishing in March 2021 [5]. Furthermore, ASEAN countries are collaborating to promote information disclosure in the fisheries industry through ship monitoring systems as well as electronic reporting and surveillance systems.

The application of technology itself has become one of the strategies used to prevent crime in a variety of industries, with computer vision being one example. Computer vision is an artificial intelligence (AI) field that enables computers to see and comprehend crucial information in photos, movies, and other visual inputs [6]. Various approaches to utilizing computer vision in various sectors of life have been developed up to this point, for example object detection.

Object detection is a computer vision algorithm that detects objects in images or videos [7]. The object detection algorithm was discovered in 2001 by the Viola-Jones duo called VJ-Detectors, which were still manufactured manually at the time. When Alexnet was invented in 2012, object detection moved to employ automation approaches. Many deep learning-based object detection algorithms have been found since then, including YOLO (You Only Look Once). Joseph Redmon was the first to discover YOLO in 2016. The YOLO algorithm is still evolving, and there is presently a seventh version of the YOLO algorithm (YOLOv7) which is a development of the fifth version of YOLO.

YOLOv7 is the most recent version of the YOLO algorithm. The primary goal of YOLOv7 development is to reduce the number of parameters and computing processes required to build a machine learning model while maintaining detection speed and accuracy. According to published papers, YOLOv7 outperforms various previously established algorithms, including YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, and others, in terms of detection speed and accuracy [8].

Several research have been conducted to detect fish species utilizing object detection with the YOLO algorithm before YOLOv7 such as YOLOv4 and YOLOv5. For example, in 2022, the YOLOv4-Tiny algorithm was used to detect fish species in an aquarium using a Raspberry Pi 4 [9]. During the same year, researchers used YOLOv5s, YOLOv5m, and YOLOv5l to detect fish species beneath water [10]. The majority of these investigations are aimed at detecting fish that are not export commodities, such as tuna. Furthermore, because the viewing angle of the image used as a dataset is extremely close to the fish, there is a risk that the item cannot be categorized if the shape of the object does not match or the distance of the object is too great.

Given this background, this study was carried out to develop a multi-object fish-detecting system in a movie utilizing the YOLOv7 algorithm. This detection system will detect a variety of items, including humans and different sorts of fish, in the video clip captured by the ship's camera. The camera's field of vision will be pointed in one direction, which is where the fish is stored, allowing all operations, including fishing and fish storage, to be monitored. Later, the model can be modified to calculate the number of fish caught by the fishermen on the vessel, so that if there is a disparity between the number of fish detected and the number of fish reported, IUU fishing on the vessel is identifiable.

# Literature Review

## Deep Learning

Deep learning is a machine learning subfield that employs numerous layers of neural networks [11]. Deep learning is an evolution of the standard machine learning method, which still employs a feature extractor to retrieve features from an input. The deep learning algorithm already has a feature extractor in its neural networks layer, allowing the deep learning algorithm to automatically extract features from input. Deep learning's architecture is based on multilayer perceptron. The multilayer perceptron operates by stacking neurons into multiple levels known as hidden layers. The layers in the hidden layer are linked together via weighted connections. A multilayer perceptron's major components are:

1. The input layer, which consists of features owned by input data.

2. Hidden layers, which are layers made up of neurons that extract feature vectors to be processed in the input layer.

3. Weight connections (edges), which is the value contained in each association between nodes in neural networks to reflect their importance in the prediction outcomes.

4. The output layer, which is the outcome of hidden layer processing and might take the shape of an actual value or a collection of probabilities.

Deep learning possesses three stages that must be executed continually in order to produce the best model. The first stage involves calculating the weights sum of each node and activation to provide a forecast, also known as the feedforward process [12]. The weights sum is the sum of the weight vectors in each perceptron. The weights total of each node will be multiplied by a non-linear activation function to obtain the prediction results for each input. The second stage involves calculating the error rate of the neural network prediction results in comparison to the expected output using an error function, which is also known as a loss function. The third stage involves using optimization functions such as gradient descent or other types to iteratively modify neural networks to the most optimal point, which is accomplished by calculating the derivative of the loss function against weights using a chain rule from the output layer to the input layer, also known as backpropagation.

## Object Detection

Object detection is one example of a real-world application of computer vision technology. Object detection is an image classification advancement in which object detection functions to enable interpretation devices used in computer vision systems to determine the location and number of things contained in an image. Object detection, in general, will identify the location and class of objects by dividing the image into smaller portions and giving a class to each broken part, allowing the items present in the image to be named [13]. The object detection framework is made up of four major components:

### Region Proposals

A region proposal is a section provided by the deep learning model in the form of a region of interest (RoIs) that will be further analyzed by the system. A region of interest is a section of an image that the system believes contains an object based on the objectness score. At this stage, the system will create thousands of bounding boxes for neural networks to evaluate and classify. The objectness score of each bounding box is the output of neural network analysis, which determines whether the bounding boxes are in the foreground (object) or background (non-object). If the bounding boxes pass the threshold established by the neural networks, they are categorized as foreground and advanced to the next level.

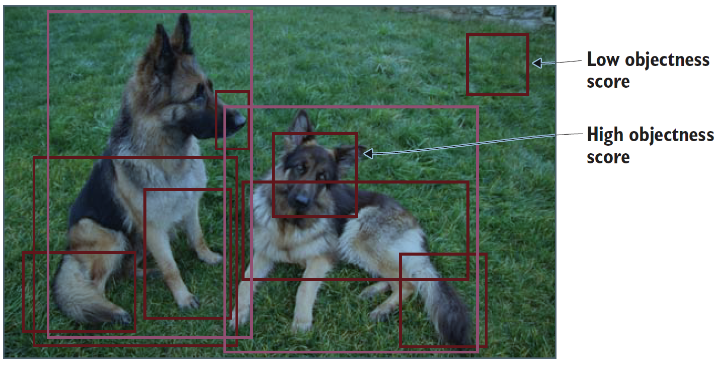


Fig. 1 Example of Region of Interest in a dog image [13]

### Feature Extraction and Network Prediction

This component is the continuation of the region proposal in which the features of the bounding boxes in the foreground region are retrieved and used to determine the class of the detected object in the image. A pre-trained image classification model will extract the characteristics in the foreground region so that the results obtained are equitably generalized. Following feature extraction, the foreground region is examined by neural networks, and two predictions are made for each foreground region, that is:

* Bounding-box prediction, which predicts the location of the foreground region in an image. The forecast will be in the form of a tuple (x,y,w,h), where x and y are the coordinates of the foreground region's center point, and w and h are the region's length and width.
* Class prediction, which predicts the likelihood of each class trained for the object in the image.

### Non-maximum Suppression (NMS)

Non-maximum suppression (NMS) is a component that ensures that each item has just one bounding box. The component works by searching all bounding boxes that point to an object for those with the highest prediction probability and eliminating those with a lower prediction probability. Here is how non-maximum suppression works in determining the highest prediction probability:

* Remove all bounding boxes whose prediction probabilities do not meet the specified confidence threshold.
* Looks through all of the remaining bounding boxes and chooses the ones with the highest probabilities.
* Calculates and compares the intersection regions between the expected bounding boxes and the ground truth bounding boxes. The value of the comparison is known as intersection over union (IoU).
* Remove bounding boxes with IoU values less than the threshold.

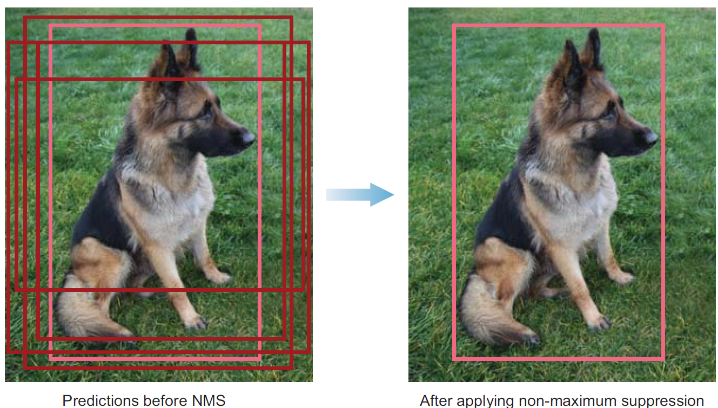


Fig. 2 Prediction results before and after applying non-maximum suppression [13].

### Evaluation Metrics

This component will examine the prediction outcomes from the previous three components. These units are frequently used in evaluating the prediction outcomes of an object detection model:

* *Frames Per Second (FPS)*

Frames Per Second is an evaluation unit that assesses how quickly an object detection model detects objects. The number of frames per second (FPS) can be estimated using the inference time value and the equation:

|  |  |
| --- | --- |
|  | (1) |

* *Intersection over Union (IoU)*

Intersection over Union is an assessment unit that will evaluate the overlap between the expected and ground truth bounding boxes. IoU will assess if the detection result is valid (True Positive) or not (False Positive) using a value range of 0-1, with the bigger number indicating a better detection result. The IoU value of a bounding box can be calculated using the following equation:

|  |  |
| --- | --- |
|  | (2) |

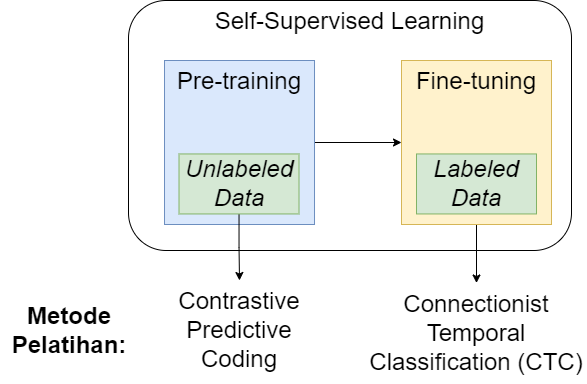
* *Precision-Recall Curve (PR-Curve)*

Precision-recall Curve is a comparison curve for the precision and recall of an object detection model that will be processed for each confidence level. Precision is an evaluation unit that quantifies a model's ability to identify a relevant object in an image. The precision of an object detection model can be calculated using the following equation:

* *Mean Average Precision (mAP)*
* *F1-Score*

## Convolutional Neural Networks

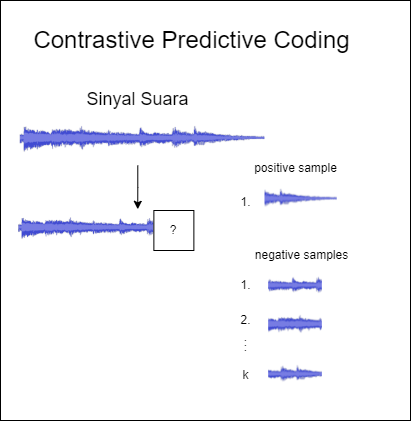
Commonly, there are 3 types of learning methods employed by machine learning models which are supervised learning, unsupervised learning, and semi-supervised learning, which depend on the type of input that is learned by the algorithm. More recently, a new learning method for machine learning models called self-supervised learning which combines large amounts of unlabeled data to learn initial representations of the input, then uses fewer labeled data to learn new representations from the input within the context of the task. The main purpose of this method is to reduce the magnitude of labeled data needed to train the machine learning model, without greatly reducing the accuracy of the model.



1. The self-supervised learning process for ASR

Fig. 3 illustrates the process of self-supervised learning that is performed for a speech recognition task. In this process, there are two steps involved, pre-training and fine-tuning. First, in the pre-training step, the model that learns through self-supervised learning is given unlabeled data, such as sound signals for ASR, to learn and extract the features within the inputs and generate the model’s initial representation of the data. In this step, the model updates its parameters by learning features from the data. One of the methods to update the model’s parameters in the pre-training step is Contrastive Predictive Coding (CPC) [10].

Contrastive Predictive Coding works by hiding some parts of the input data, then the model has to predicts and replace the missing inputs given the data previous to the missing pieces. The input data can be of many forms, such as hiding one area from an image, one word from a sentence, or one part of a speech waveform in the context of an ASR model. The CPC process happens at the pre-training step of the self-supervised learning method, therefore only needs the input data itself without the label that is given to represent the data.

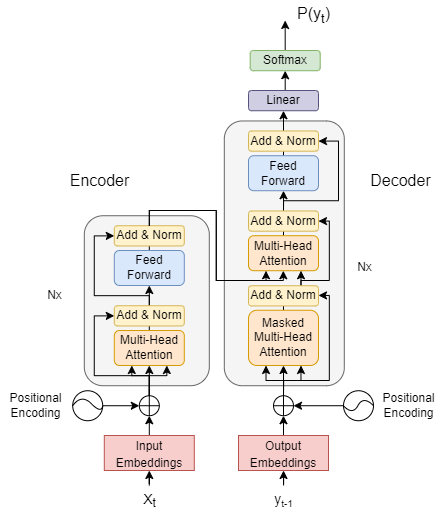


1. Automatic speech recognition system prototype

Fig. 4 illustrates the Contrastive Predictive Coding process that is performed by an ASR model that employs the self-supervised learning method. First, there is a sound signal (waveform) that is fed into the model which its ending is then masked for a set duration. The masked part of the waveform is called a positive sample. Aside from the positive samples, the model also extracts other waveforms of the same duration from different parts of the input as negative samples. A self-supervised learning model then solves a classification task of differentiating the positive samples from a number of other negative samples according to the data given before the masked waveform. If the self-supervised learning model can correctly classify the positive sample from the other negative samples, then it can be inferred that the model has learned a general representation of the features extracted from the data even without given a set label that describes the data.

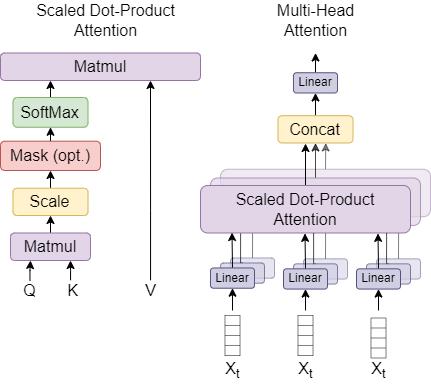
## YOLOv7

Transformer Networks, or commonly known as the Transformer, is one of the more recent Deep Learning models, first introduced in 2017 by [11]. The Transformer is a sequence-to-sequence model which accepts an input sequence, such as a sequence of words or sound signals, and outputs another sequence based on the model’s prediction. The Transformer commonly uses the supervised learning method, however new researches show that other learning methods such as self-supervised learning can be employed [12].



1. General architecture of the Transformer [11]

The architecture of the Transformer can be seen on Fig. 5, which is divided into 2 main parts, the encoder and decoder. The encoder learns the input data and converts it into a representation of the data. The decoder then processes the representations resulted by the encoder and learns its relationships to the desired output of the Transformer, which is the labels given to solve the machine learning task. This encoder-decoder architecture can be stacked to a number of blocks within the Transformer which extracts and learns different features from the input data, and opens up the possibility for more optimal use of parallelization which could not be done for other sequence-to-sequence models [11].

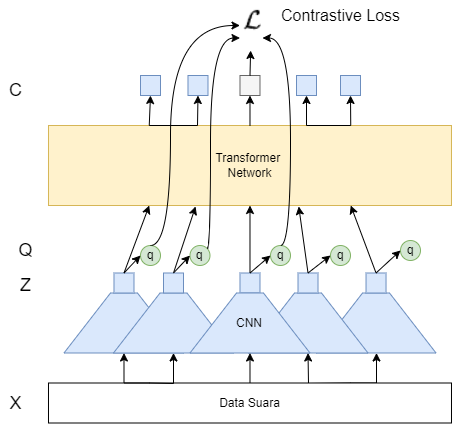


1. Architecture of Multi-Head Attention in Transformer [11]

The main method of the Transformer to learn representations of the input data to generate a new sequence is by way of the Attention Mechanism. The Attention Mechanism is used within the Multi-Head Attention component of the Transformer that is placed both within the encoder and decoder. The Attention Mechanism works by mapping its inputs to 3 vectors which are the query, key, and value through a Linear Layer. These query, key, and values are used to compare different inputs and learn the relationships within them, along with the relationship between inputs at one timestep of the sequence with its output. These vector values are learned by the network to create different representations of the input to be used for generating the Transformer’s output sequence. The Attention Mechanism can also be stacked in a number of Attention Heads, which also learn different relationships in the data.

## Polygon Annotation

The wav2vec 2.0 Deep Learning model, first introduced by Facebook AI in 2020 [5]. This model learns speech representations from sound signals through a self-supervised learning method, to be used for a sequence-to-sequence task such as automatic speech recognition. One of the main features of this model is its capability to learn speech representations from a low amount of labeled data, which is ideal to use in automatic speech recognition systems for low-resource languages. In the paper, this model yields a WER of 4.2/8.2 using the LibriSpeech dataset for English language with only 10 minutes of labeled data given [5].



1. Architecture of wav2vec 2.0 [5]

The architecture of the Transformer consists of two main parts, the feature extractor which creates vector representations of sound signals through a Convolutional Neural Network, then decodes those representations against a set number of labels in a Transformer Network. This model takes inputs of raw speech waveforms or sound signal data, and outputs a sequence of letters which makes up different words to form sentences. This model learns through the self-supervised learning method, where it is given raw speech waveform data during the pre-training step, then given sound signal data along with its spoken transcript in text form during the fine-tuning step.

There are multiple implementations of the wav2vec 2.0 architecture which creates different variants of the wav2vec 2.0 model. These implementations depend on the number of layers and dimensions used for the Transformer network of the model. The wav2vec 2.0 BASE variant uses 12 Transformer blocks, dimensions of 768 internally and 3072 for the model, and 8 attention heads whereas another variant, the wav2vec 2.0 LARGE uses 24 Transformer blocks, dimensions of 1024 internally and 4096 for the model, along with 16 attention heads. One implementation of the wav2vec 2.0 model, called wav2vec 2.0 XLSR-53, uses the wav2vec 2.0 LARGE model architecture and is already pre-trained with 56000 hours of unlabeled sound signal data from 53 languages [13]. The wav2vec 2.0 XLSR-53 model can then be used in an ASR system by performing the fine-tuning step by feeding sound signal data with its transcript for the language to be recognized.

## CommonVoice

The CommonVoice dataset is a dataset which provides sound signal data of speech utterances for a variety of languages, collected from different speakers all over the world [14]. Along with the sound signal data, the dataset also provides additional information such as the text transcript spoken in the sound signal, along with information about its speaker for each sample. The CommonVoice dataset has many versions and is updated to a new version with the addition of new data. One of the languages provided by this collective dataset is Indonesian, which in the 8.0 version contains around 60000 samples of audio data as a waveform format which equals to about 50 hours of speech.

1. Splits of Data Within the CommonVoice 8.0 Dataset for Indonesian Language

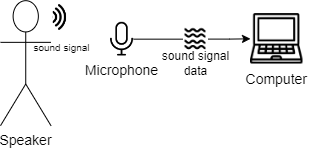
| Name of Split | Number of Samples |
| --- | --- |
| train | 5302 |
| dev | 3207 |
| test | 3608 |
| validated | 22874 |
| other | 22385 |
| invalidated | 2442 |
| Total | 59818 |

The CommonVoice dataset is divided into many splits which can be used for training, testing, and evaluating a machine learning model. TABLE I. shows the amount of data within each split of the CommonVoice 8.0 dataset for Indonesian language.

# Methodology

## System Architecture and Design

From the prototype of the automatic speech recognition system that is implemented, there is one computer that is connected to one microphone. This system receives sound signals of human speech utterances from the microphone, then outputs the text equivalent of the same sentences that was said by speaker. This is possible due to the ASR model based on Deep Neural Network that is running as a process within the computer, which uses the sound signals received by the microphone as input to the speech to text model. This system is visualized in Fig. 8.

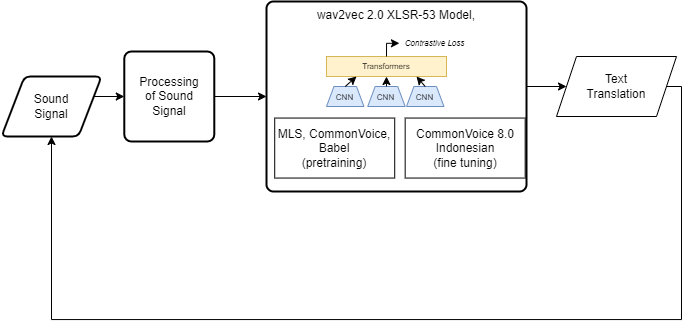


1. Automatic speech recognition system prototype

In this system, the computer runs a process that is a Python program which processes the sound signal data and gives its text output from a machine learning model. The model works by representing parts of the speech signal sound signals from the speaker as vector representations that are then translated to different characters that makes up a word. After processing the speech signal data given by the microphone to the computer into vector representations, the Python program then outputs the words that were uttered by the speaker within the sound signal as sentences in the form of written text.

## Speech Recognition Model Design

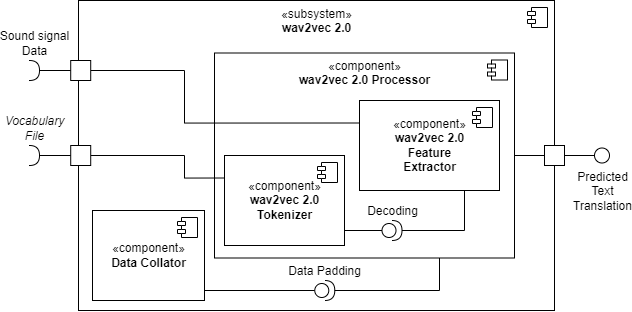
The automatic speech recognition machine learning model designed accepts input of sound signals that which are also processed further before given to the model. For the design of an automatic speech recognition system in this research, the wav2vec 2.0 XLSR-53 model is used. After the sound signal is processed and analyzed by the model, it will then output the text equivalent of the words uttered by the speech inside the sound signal.



1. Schema of Speech Recognition Model

Fig. 9 illustrates a schema of the speech recognition model using the wav2vec 2.0 model from the design. First, the sound signal is processed before it can be used as input for the wav2vec 2.0 model. The variant of the wav2vec 2.0 model used in this research is the wav2vec 2.0 XLSR-53 variant which has the same architecture as wav2vec 2.0 Large model, but uses the speech data from various languages in the MLS, Babel, and CommonVoice datasets during pretraining. After initial processing, the model classifies different parts of the sound signal to different characters based on the sound that is present within the sound signal then combines these characters to form a word. These words are then combined to form different sentences. The final output of the wav2vec 2.0 XLSR-53 model is the text translation in written text form derived from the sound signal used as input for the model.

As a speech recognition model, the wav2vec 2.0 model implemented can be split into different components, as outlined in Fig. 10.

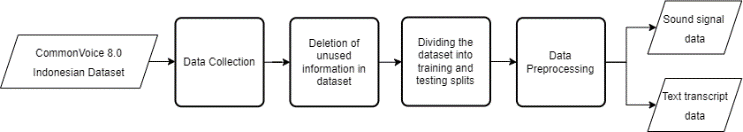


1. Schema of the components inside a wav2vec 2.0 model

Fig. 10 illustrates the different components inside a wav2vec 2.0 model, which consists of a wav2vec 2.0 Processor and a Data Collator. The wav2vec 2.0 Processor functions to process the input data of sound signals given to the model then outputs the predicted translation from the recognized speech in text form. This component itself consists of two sub-components, the wav2vec 2.0 Feature Extractor, which is used to remodel the input sound signal into its vector representation, and the wav2vec 2.0 Tokenizer which classifies the vector representation values of the sound signals into different letters or sounds that make up a single word. In the wav2vec 2.0 implementation designed for this research, a Data Collator is also used, which functions to add padding or elongates certain sound signals with a padding symbol, so that all of the sound signals within one batch have the same length or duration, making it easier and faster for the model to process the batch.

## Data Preparation

Before the speech recognition model is given sound signals along with its transcripts during training, there are a few steps that must be passed to prepare said data. First, the data is collected from the CommonVoice 8.0 dataset for Indonesian language. This dataset consists of sound signals from speech spoken by different Indonesian speakers, along with the actual text transcript each corresponding sound signal. Aside from speech data and its transcripts, in the CommonVoice dataset there are additional information about the speaker for each sound signal and transcript pair, which is data that is not needed for training an ASR model. Therefore, this data is deleted, or in other words unused in the next step of the data preparation pipeline. After that, the next step in the data preparation process consists of splitting the training and testing datasets to train and evaluate the model, along with performing data preprocessing so that the data is uniform in shape and type with what the ASR model needs. The data preparation process is visualized in Fig. 11.



1. Data Preparation Steps

Fig. 11 illustrates the steps needed to prepare the dataset before it can be used in training and evaluation of the ASR model. From these preparation steps, there are two data types required as inputs for the ASR model. First the data that is required to train, evaluate, and test the ASR model based on wav2vec 2.0 consists of sound signals that are derived from an audio file of type MP3 or WAV. To produce the optimal result from the model, the sound signals used should be clips of speech utterances from different speakers, each with a duration of 5 to 30 seconds. The sound signals that are prepared will be used as input so the model can predict the text translation equivalent to that sound signal, consisting of different words which make up the utterance said by each speaker.

During the fine-tuning step of training, labeled data or the actual transcripts of each sound signal is needed by the model. The purpose of using labeled data is to eventually build the most optimal model, the model must adjust its parameters such as weight values within it, by calculating the difference between the text transcripts predicted by the model with the actual text transcripts from the dataset. Knowing this purpose, during training the model must be given labeled data which are pairs of sounds signals that are speech utterances by different speakers with its corresponding transcript that notes the words spoken in each utterance.

This research uses the Datasets library made by the HuggingFace team to access the Common Voice 8.0 dataset of Indonesian language [15]. This library eases the data collection for datasets which are commonly used, such as CommonVoice. It accomplishes this by first downloading the dataset to be used and caching in a folder within the computer’s file system, therefore removing the need to redownload the dataset for future uses even for different programs. To download a dataset and cache it for later use, this library exposes the load\_dataset function, as seen in this example.

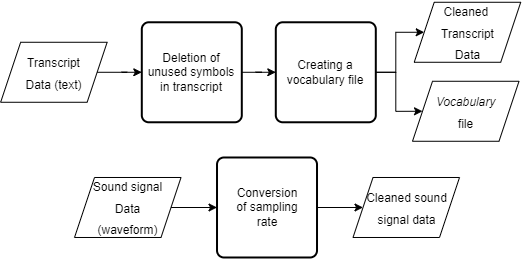
train\_dataset **=** load\_dataset**(**"common\_voice"**,** "id"**,** split**=**"train+validation"**)**

eval\_dataset **=** load\_dataset**(**"common\_voice"**,** "id"**,** split**=**"test"**)**

1. Code snippet to collect data from the Datasets library

Fig. 12 shows a code snippet that functions to save the CommonVoice dataset for Indonesian language with the load\_dataset function given by the Datasets library. The same function can also be used to divide different subsets of the dataset into splits for training and testing the model, by using the split parameter that dictates which subset of the CommonVoice dataset used for each split.

Before each pair of sound signals and its transcript can be used for the ASR model, a preprocessing step must be done in advance so that the shape and type of both data types are uniform with what the model needs for its inputs. There are a few preprocessing steps required, which are different for the sound signal (represented as waveform) and transcript data (represented as text). These preprocessing steps are outlined in Fig. 13.



1. Preprocessing steps for the data

Fig. 13 illustrates the preprocessing steps for the data used in the wav2vec 2.0 model. For the transcript data which is in text form, symbols that are present in the text such as punctuation marks, subscripts, and symbols rarely used in Indonesian language are removed from the text or replaced by a letter from the alphabet. The next step consists of creating a vocabulary file which represents the different characters that can be used to classify a spoken character from the sound signal. For the sound signal itself, a conversion of sampling rate is done to reduce the dimension of the data for faster processing by the model.

**def** remove\_special\_characters**(**batch**):**

batch**[**"sentence"**]** **=** re**.**sub**(**chars\_to\_ignore\_regex**,**

''**,**

batch**[**"sentence"**])**

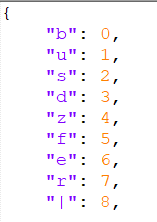
**.**lower**()** **+** " "

**return** batch

1. A code snippet to remove any unnecessary symbols or characters from the transcript data

The code snippet in Fig. 14 shows the removal of different symbols which are not from the alphabet and unnecessary whitespace from the speech transcript data used for the model. Regular Expression (Regex) is used to find and remove the different symbols which are present in the text with a space or another letter from the alphabet for each transcript data in the dataset. A few examples of the symbols removed during this process is any punctuation marks and letters with accent symbols.

Within these preprocessing steps for the transcript data, there is a step of extracting features to generate a Vocabulary File.



1. An example of content within the Vocabulary File

Fig. 15 shows an example of a Vocabulary File that will be used in the decoding step of the wav2vec 2.0 model. This file consists of each character after the removal of unnecessary symbols within the previous step, which mostly leaves the 26 letters of the alphabet. Aside from these letters, there are also a few directives or references which can be used by the model to better understand the context of speech within the sound signal data. These directives are the “|” symbol which replaces any spaces within the transcript to better differentiate it from “UNK” symbol used by the model for speech which could not be recognized, and the “PAD” symbol which signifies a pause within the speech. The generated Vocabulary File is then stored within the computer, so that it can be used by the model at any time during training, testing, or inference usage. The order of each letter and directive within the Vocabulary File does not matter and will be read the same every time by the model’s tokenizer.

train\_dataset = train\_dataset.cast\_column("audio", Audio(sampling\_rate=16\_000))

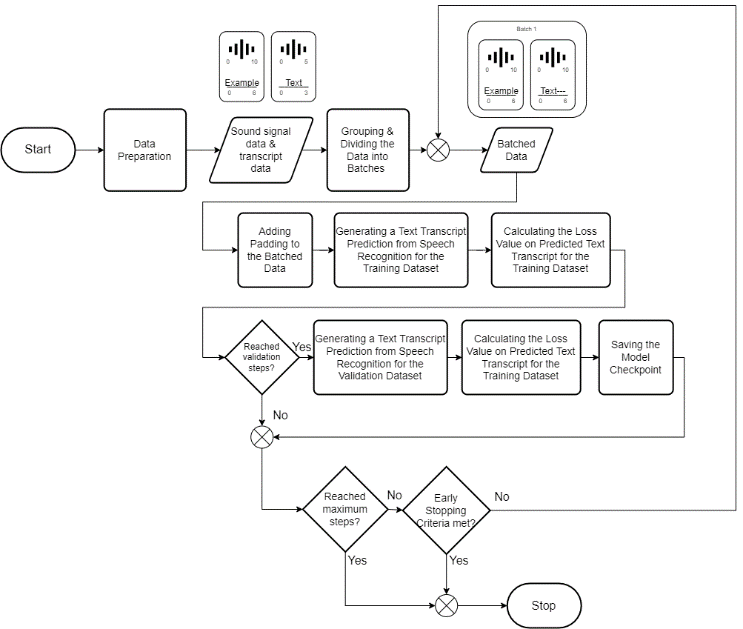
eval\_dataset = eval\_dataset.cast\_column("audio", Audio(sampling\_rate=16\_000))

1. A code snippet to convert the sampling rate of the data

Fig. 16 shows the process to convert the sampling rate of the sound signal data using the cast\_column functions provided by the Datasets library. The conversion of sampling rate is the only preprocessing step that is done on the sound signal data, which is done to reduce the dimension of said data so that the model can process it easier and quicker. In the CommonVoice 8.0 dataset for Indonesian language, a sampling rate value of 48 kHz is used to capture the original continuous sound signal, which in this preprocessing step is converted to 16 kHz or only sampling one of three data points from the original signal.

## Design of the Training Process for the Speech Recognition Model

After finishing the data preparation steps with steps such as collection and preprocessing of data, the data produced should consist of pairs of sound signals with its transcript, both with the correct shape and type needed by the model. If so, then the data is ready to be fed for the model’s training process. An overview of this training process consists of grouping the data into batches, then using the sound signal data to generate a predicted transcript text for each batch and calculating the loss or the difference between the transcript text predicted by the model from the actual transcript text for each sound signal. Then, the model undergoes a validation process after a set number of steps or an amount of training data, to determine which model will be stored in the computer depending on its performance. Both the training and validation process are conducted several more times in a loop as illustrated in Fig. 17, so long as the model hasn’t reached its maximum steps or the early stopping criteria isn’t met.



1. Schema of the training process for the wav2vec 2.0 model

Fig. 17 shows the training process for the wav2vec 2.0 model that is conducted multiple iterations in a loop for the given input data, starting from the grouping and dividing the data into batches depending on the batch size parameter that is set during the training process. A larger batch size can significantly speed up the training process, but requires the wav2vec 2.0 model to be loaded into the Graphics Processing Unit (GPU), therefore using more Virtual RAM from the GPU. For every batch, padding is given to equalize the length of every sound signal data for each batch, so that no one signal can influence the model too much in one iteration. After the model has finished the training process for one batch, then it denotes the end of one step. Then, the training process is conducted from the start using another batch in the next step.

The training process for the wav2vec 2.0 model consists of two phases, that is the training phase and the validation phase, which correlates to the datasets used in each of the phases. In the training phase, a subset the CommonVoice 8.0 dataset for Indonesian language is used, consisting of only the train and dev splits (about 30% of the total data). In this phase, the model processes the data then predicts the text translation from the sound signal of speech utterances, then calculates the loss given by that prediction in relation to the actual text transcript from the dataset. The function of this phase is to find the weight values that are most precise for the model to perform speech recognition and generate the corresponding transcript text. The change of weights is dictated by the learning rate hyperparameter that is set before the training process. A learning rate that is too large will result in the weights of the wav2vec 2.0 model having frequent large changes and become too unstable, which leads to not finding the optimal weight values that produce the highest accuracy for the model. Conversely, a learning rate that is too small will result in very slow training times due to the miniscule changes that happen to the weight values during each iteration.

The second phase of this training process for the wav2vec 2.0 model, the validation phase, is not conducted on each step, but only after a certain number of steps. This phase uses a different subset of the CommonVoice 8.0 dataset for the Indonesian language, which is the test split (14% of total data). The validation phase is used to check on the weights that are set during the training phase for a dataset which is not learned by the model before, by calculating the loss from the transcript predictions from sound signals given by the new dataset. Aside from that, this phase is also used to calculate the overall WER to determine the overall performance of the current wav2vec 2.0 model when performing speech recognition to generate the text transcript given a sound signal. If the performance of the model is better than the performance from the last validation phase, then the model will be stored in the computer so that it can be used for an ASR system.

The wav2vec 2.0 model’s training process is repeated as long as two conditions are met. First, there is a maximum number of steps that is set and applied for the training process. The model will automatically end the training process if the number of training steps have exceeded the set maximum number. Aside from that, in the design of the training process for the wav2vec 2.0 model, early stopping is implemented to help stop the training process even if the maximum number of training steps haven’t been reached. Early stopping is used to prevent the model from overfitting, where the model has really good performance when given data included in the training process, but gives poor performance when given data that the model hasn’t learned before. Overfitting is caused by learning the training data for too long (for too many steps) during the training process. Early stopping helps stop the model when it has detected minimal decreases in the model’s loss after a set number of steps, which signals that the model has trained enough from the training data.

## Design of the Evaluation Process for the Speech Recognition Model

The planned scenario used to evaluate the wav2vec 2.0 model consists of loading the model into the GPU, then performing an inference test by predicting each sound signal from the test split of the CommonVoice 8.0 dataset for Indonesian language. From this prediction outputs a vector of values that is then decoded by the model into text form, similar to the actual transcript from the dataset. Each of the predicted text is then compared to the actual transcript from the dataset. Lastly, to determine the model’s performance, the overall WER is calculated from the comparison of predicted transcript and the actual transcript from the dataset by calculating the WER for each sentence and computing the average from the dataset.

# Simulation Results

## System Implementation

The system to be implemented, evaluated, and optimized consists of the wav2vec 2.0 XLSR-53 that is further trained through the fine-tuning process with data from the CommonVoice 8.0 dataset for Indonesian language. First, the wav2vec 2.0 model is compared to another Deep Neural Network model to gain perspective as to how the wav2vec 2.0 model performs in recognizing speech of Indonesian language. Due to the constraints from the hardware used for this research, the batch size set for training and testing the model is 4, as the largest value that can be processed by the GPU.

For the comparison, this research uses a Deep Neural Network model that is commonly used in ASR systems, a Bidirectional Long Short-Term Memory (LSTM) that uses a Convolutional Neural Network (CNN) to perform feature extraction [16]. The Bidirectional CNN-LSTM model slightly differs from the wav2vec 2.0 model, where the CNN-LSTM model doesn’t process raw sound signal data as waveforms, but uses Mel-Frequency Cepstral Coefficients (MFCC) to generate 40 feature input vectors that each corresponds to 25 milliseconds of sound signal data. The architecture of this model consists of 3 Convolutional Layers, each with 32x32 input and output dimensions, kernel size (3, 3, 3), stride (3, 11, 1) and are arranged consecutively as a Convolutional Neural Network to further extract the features generated by MFCC. Once feature extraction is applied, the vector representation of sound signal features is fed into the LSTM network which consists of 512 hidden neurons. The output of this LSTM classifies sound signal representations into the 28 possible labels, with the composition of 26 letters, one space character, and one blank character for padding.

1. Computer Specifications

| Category | Type | Name/Version |
| --- | --- | --- |
| Hardware | Processor (CPU) | Intel Core i7-8700 @ 3.20 GHz (12 CPUs) |
| Graphics Card (GPU) | NVIDIA GeForce RTX 2080 SUPER (8GB) |
| Random Access Memory (RAM) | 32 GB DDR4 |
| Storage | 2 TB HDD |
| Software | Operating System | Ubuntu 20.04 LTS Linux |
| Python | Python 3.9.12 |
| Conda | 4.11.0 |
| PyTorch | 1.11.0 |
| CUDA Toolkit | 11.3 |
| Automatic Speech Recognition (ASR) Model | wav2vec 2.0 XLSR-53 |
| Intergrated Development Environment (IDE) | Jupyter Notebook 6.4.11 |

A computer is needed to develop the machine learning model that is trained, tested, and ran for this research. The specifications for the computer used for this research can be seen on TABLE II. Not all of the specifications for the computer used to run the model will affect the result and accuracy of the machine learning model. However, the specifications for the GPU used in the computer heavily affects how fast the model runs during training and testing. A GPU with a large amount of Virtual Memory (VRAM) can increase the batch size set that can be processed by the model in one iteration. Aside from that, the amount of CUDA Cores inside the GPU also affects how fast the model can process each sample of the dataset.

## Evaluation Scenario

There are two main scenarios used to evaluate the wav2vec 2.0 model, the first being evaluating the overall performance of the wav2vec 2.0 XLSR-53 model and how it compares to a CNN-BiLSTM model that is used in [16]. In the training process of both models, the amount of data used are the same to compare both models within the same context. After the comparison between the two models is finished, then this research also evaluates the wav2vec 2.0 XLSR-53 model internally, by comparing different configurations of the wav2vec 2.0 model which results the best accuracy in the context of recognizing speech for the Indonesian language. For all of the experiments done in each evaluation scenario, the dataset used is the test split from the CommonVoice 8.0 dataset for Indonesian language. Aside from that, all experiments are repeated 5 times, with the average and standard deviation of those experiments are used as the result of each experiment, to make sure that the result is repeatable and consistent.

There are two main criteria used to evaluate and compare the wav2vec XLSR-53 model with the CNN-BiLSTM model. The first and most important criteria is the WER that illustrates the performance of both models when performing inference and generating text from a sound signal. The WER of both models needs to be evaluated because to eventually implement the model in an ASR system, the system must yield high accuracy from the speech to text process, which is equivalent to a low WER value. When evaluating this criterion, how much data used to train each model needs to be noted, because it affects how well the model performs when learning a low-resource language such as Indonesian.

The second criterion that is used to evaluate both models is the size of each model when loaded into the GPU, or how much memory from the GPU is allocated to prepare and execute each model along with the storage required to save the model in the computer. The model that requires higher memory allocation raises the minimum specifications for the computer to execute the model in an ASR system, especially for the GPU. This then also raises the cost requirement to execute the offline ASR system and makes it harder for implementation in everyday household items. Because of this reason, it is important to evaluate the GPU memory usage when comparing both models.

When evaluating the wav2vec 2.0 model internally, the criterion that is evaluated are the variation of learning rate configurations and the amount of data used during training. Evaluation of the model’s learning rate is conducted due to tuning of learning rate being an important process to develop a Deep Neural Network model that not only correctly learns the data given as input but can also do so in a short amount of time. For this evaluation, this research conducts experiments of learning rate with values and to determine the correct scale for the learning rate value. Once the correct scale for the learning rate is determined, then the model can be optimized further by changing the learning rate multiplier. Learning rates of multipliers 1, 3, and 5 are used. Aside from learning rate, another experiment is conducted which varies the amount of data used during training to determine how far the performance of the wav2vec 2.0 model can be improved along with the increase of data available during training. In this experiment, the number of sound signal and text transcript pairs used amount to about 8000, 16000, and 32000 of samples. The metric evaluated during this experiment is the WER value for all 3 variations of amount of data used during training the model with and without the addition of a Language Model (LM).

## Evaluation and Comparison of the wav2vec 2.0 Model

From the scenarios that are mentioned in the previous chapters concerning the evaluating and comparison of the wav2vec model, there are two experiments conducted which tests both the wav2vec 2.0 XLSR-53 and CNN-BiLSTM models with and without a Language Model.

In the first experiment, the wav2vec 2.0 model is compared against the CNN-BiLSTM model used as a baseline. During this experiment, there are a few hyperparameters set for the wav2vec 2.0 model such as the model’s learning rate set at and batch size at 4. During the training process, the wav2vec 2.0 model learns from labeled data pairs of sound signals and text transcript data from 8.000 samples taken from the train and dev splits of the CommonVoice 8.0 dataset for Indonesian language. The CNN-BiLSTM models is also trained using the same amount of data, along with the addition of samples from the other split in the CommonVoice 8.0 dataset which results in an estimated 50.000 samples of pairs of sound signals and their equivalent text transcript. This purpose of these experiments is to show and compare the performance of the wav2vec 2.0 model in learning speech recognition from a low resource language to another model which in the past requires a larger magnitude amount of data. The trial which uses 50.000 samples of sound signal and text transcript data is not conducted for the wav2vec 2.0 model, due to the Virtual RAM constraints of the GPU that could not accommodate the high memory requirements of the wav2vec 2.0 model. Below are the results from this experiment:

1. The WER Result From the wav2vec 2.0 XLSR-53 and CNN-BiLSTM Models

| Model | Amount of Labeled Data Samples | WER |
| --- | --- | --- |
| CNN-BiLSTM | 8.000 |  |
| 50.000 |  |
| wav2vec 2.0 XLSR-53 | 8.000 |  |
| 50.000 | - |

The results from TABLE III. shows that the wav2vec 2.0 XLSR-53 model significantly outperforms the CNN-BiLSTM model commonly used in previous ASR systems. The wav2vec 2.0 model yields a WER of 25,65%, a difference of 30% against the CNN-BiLSTM model, but requires a lot less amount of labeled data. When both models are compared with the same amount of labeled data, then the results clearly show that the wav2vec 2.0 model also significantly outperforms the CNN-BiLSTM model which yields a WER of 93%. The results from the CNN-BiLSTM models when trained with 8000 samples of labeled data are too high to be integrated in an ASR model and will most likely result in poor performance for the overall system.

The ability of the wav2vec 2.0 model to recognize speech even when trained with a small amount of labeled data stems from the pretraining step during the model’s self-supervised learning process which enables the model to learn the general representations of speech before the model is further trained with the fine-tuning step. Because the model can learn general speech representations during the pretraining step, then the model only needs to adjust the parameters to better understand the sound signals depending on the data during the fine-tuning step. In comparison, a traditional approach such as the CNN-BiLSTM model needs to be fully trained with all of the available labeled data or from a transfer learning process, both of which require a large amount of labeled data.

However, TABLE IV. shows the flaw of the wav2vec 2.0 model lies in the computational resources required to execute the model, which also caused the trial of training the wav2vec 2.0 model with 50.000 samples of labeled data could not be conducted during this research, due to the limitation from the computer specifications. To better understand this requirement, an experiment is further conducted that evaluates 2 metrics from both models, which are the VRAM usage during training and during execution along with the amount of storage used by both models when stored as a PyTorch file.

1. The Computational Requirements From the wav2vec 2.0 XLSR-53 and CNN-BiLSTM Models

| Model | VRAM Usage (MB) | | Storage Usage (MB) |
| --- | --- | --- | --- |
| During Inference | During Training |
| CNN-BiLSTM | 1203 | 2723 | 136 |
| wav2vec 2.0 XLSR-53 | 2097 | 7643 | 1300 |

TABLE IV. illustrates that the wav2vec 2.0 XLSR-53 model requires a high amount of VRAM and storage, which makes it not ideal to train and deploy in hardware with minimal computational and storages resources. During both inference and training, the wav2vec 2.0 model requires at least twice the amount of VRAM when compared against the CNN-BiLSTM model. The wav2vec 2.0 model also requires tenfold larger storage. Both of these factors are caused by the complexity of the wav2vec 2.0 architecture which uses a lot more layers and higher dimensionality that results in a large number of weight and bias parameters. The use of the Transformer model architecture in wav2vec 2.0 also requires higher computational resources when compared to other autoregressive models such as an LSTM, due to the Transformer model using the GPU’s parallelization capability to a higher extent.

Aside from evaluating and comparing the performance of the wav2vec 2.0 model with the CNN-BiLSTM models, this research also experiments with the addition of a Language Model to improve the performance of both machine learning models. A Language Model can be used to improve the performance of an acoustic model such as a Deep Neural Network, by computing the subsequent words that can appear in a sentence and correcting any misspelling generated during the model’s decoding step. The language model used in this experiment is a 2-gram Language Model derived from all of the text transcripts inside the CommonVoice 8.0 dataset for Indonesian language, created by using the KenLM software. Even though the use of the same datasets for training an Acoustic Model and Language Model may result in higher WER due to the fact that the words found in the speech utterances are also found in the language model, the addition of a Language Model is evaluated to see how it improves the Acoustic Model’s performance.

1. The WER Result From the wav2vec 2.0 XLSR-53 and CNN-BiLSTM Models with a Language Model

| Model | Amount of Labeled Data Samples | WER |
| --- | --- | --- |
| CNN-BiLSTM | 8.000 |  |
| 50.000 |  |
| wav2vec 2.0 XLSR-53 | 8.000 |  |
| 50.000 | - |

The results in TABLE V. Show that the addition of a Language Model decreases the WER yielded by both the wav2vec 2.0 XLSR-53 and CNN-BiLSTM models, no matter the amount of data used during training. Aside from that, the Language Model also decreases the standard deviation of both models for all trials during the experiment, which means a more consistent result from the decoding process. This is caused by the LM which contains the probabilities that represent the relationship of subsequent words being a constant value for all the experiments, even if the parameters of the Acoustic Models change. The results shown from this experiment are still aligned with previous results, with the wav2vec 2.0 model outperforming the CNN-BiLSTM model that yields WER of 13,64% and 21,66% WER respectively, while requiring a smaller amount of labeled data.

## Evaluating the wav2vec 2.0 Model Internally

The first experiment used to evaluate the internal configuration of the wav2vec 2.0 XLSR-53 model is the variation of learning rate values used during the training process.

1. The WER Result From the wav2vec 2.0 XLSR-53 with Different Scales of Learning Rate

| Learning Rate Scales | WER |
| --- | --- |
|  |  |
|  |  |
|  |  |

The results in TABLE VI. show that the wav2vec 2.0 XLSR-53 model yields the lowest WER of 40.2% when training the model for 10 epochs using the scale for its learning rate. When compared to different scales, the results show that the scale also enables the model to learn from the data, but the larger learning rate causes the model to never come to its global optimum therefore resulting in poor performance. This causality of the model being unstable can also be seen from the high standard deviation during which equates. On the contrary, the scale causes the model to undergo a very slow training process which results in minimal change to its weight values and only yielded a WER of 98.55% during evaluation. From these results, it is clear that the use of scale for the model’s learning rate enables the model to successfully learn the given input data in a timely manner. After this scale is determined, then the next step is to vary the different learning rate multipliers, such as , , and .

1. The WER Result From the wav2vec 2.0 XLSR-53 with Different Learning Rate Multipliers

| Learning Rate Scales | WER |
| --- | --- |
|  |  |
|  |  |
|  |  |

The results in TABLE VII. show that the increase of learning rate multipliers does not equal to a more optimal result. This is caused by the fact that and multipliers are too large especially after the wav2vec 2.0 model undergoes many steps during the training process. When that occurs, then the model only needs a slight adjustment in its parameters to reach the global optimum, but the larger multipliers cause the model to change its parameters to a greater extent and results in a higher WER.

For the final experiment, this research evaluates different amounts of data samples (pairs of sound signal data and its equivalent text transcript) used during the training process for the wav2vec 2.0 XLSR-53 model. This experiment uses WER for the metric to identify the performance of the wav2vec 2.0 model (with and without an additional Language Model) with respect to different amounts of labeled data. The language model used is the same one used in previous, derived from all of the text transcripts from the CommonVoice 8.0 dataset for Indonesian language, created using KenLM.

1. The WER Result From the wav2vec 2.0 XLSR-53 with a Variation of Amount of Labeled Data Used During the Training Process

| Amount of Data Samples Used During the Training Process | WER without LM | WER with LM |
| --- | --- | --- |
| 8239 |  |  |
| 16533 |  |  |
| 33066 |  |  |

TABLE VIII. shows the variation of amount of labeled data samples of Indonesian speech utterances used during the training process in the training step of the wav2vec 2.0 XLSR-53 model. The results show that the increasing amount of labeled data used during training has a positive effect on the model’s performance. The more data used, the lesser the WER yielded by the model. From the 3 varying amounts of labeled data used, a small increase can be seen from roughly ~8000 to ~16000 samples which yields 25.65% and 23.46% respectively without a Language Model. A more significant WER decrease can be seen when using ~32000 samples yielding a WER of 15.6%, the lowest of all 3 variations. The same pattern can be seen when the model performs speech recognition with a Language Model, which shows that the performance of the wav2vec 2.0 XLSR-53 model can be improved with respect to the amount of labeled data used during the training process.

# Conclusion

From the research that is done in this paper, it is concluded that the wav2vec 2.0 XLSR-53 model can be implemented in an offline automatic speech recognition system that still performs well, yielding a WER of 25,96% when tested with the test split of the CommonVoice 8.0 dataset for Indonesian language. This model outperforms traditional approaches such as a CNN and Bidirectional LSTM model while requiring a significantly lesser amount of labeled data. This helps to create an ASR model that also performs well on low-resource languages, such as Indonesian. However, the wav2vec 2.0 model requires higher amounts of computational resources such as Video RAM and storage than the CNN-BiLSTM, which may not be suitable for devices with low-end specification.

One of the main points that can be used for further works is to integrate the wav2vec 2.0 XLSR-53 model within an offline ASR system that can perform speech recognition and inference in real-time, followed by the deployment of such system with an easy-to-use interface. Other aspects can also be explored to improve the performance of the wav2vec 2.0 model for specific low-resource languages, such as hyperparameter tuning and data augmentation. Lastly, the effects of pretraining for different languages may also be explored, for example creating a wav2vec 2.0 model that is pretrained and fine-tuned on a dataset for the Indonesian language.

# References

|  |  |
| --- | --- |
| [1] | J. Li, *Recent Advances in End-to-End Automatic Speech Recognition,* 2020. |
| [2] | S. Suyanto, A. Arifianto, A. Sirwan and A. P. Rizaendra, "End-to-End Speech Recognition Models for a Low-Resourced Indonesian Language," in *8th International Conference on Information and Communication Technology (ICoICT)*, Yogyakarta, 2020. |
| [3] | G. Germanos, K. Dimitris, N. Kolokotronis and N. Georgious, "Privacy Issues in Voice Assistant Ecosystems," in *2020 IEEE World Congress on Services (SERVICES)*, Beijing, China, 2020. |
| [4] | Verizon, "Data Breach Investigation Report (DBIR)," Verizon, 2022. |
| [5] | A. Baevski, H. Zhou, M. Abdelrahman and M. Auli, *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,* Arxiv, 2020. |
| [6] | J. M. Levis and R. Suvorov, "Automatic Speech Recognition," in *Encyclopedia of Applied Linguistics*, Blackwell, 2012. |
| [7] | A. Dev, A. Jain and S. Bhatt, "Acoustic Modeling in Speech Recognition: A Systematic Review," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 11, no. 4, p. , 17 Maret 2020. |
| [8] | Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998. |
| [9] | S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a Convolutional Neural Network," in *2017 International Conference on Engineering and Technology (ICET)*, Antalya, 2017. |
| [10] | A. v. d. Oord, Y. Li and O. Vinyals, Representation Learning with Contrastive Predictive Coding, Arxiv, 2019. |
| [11] | A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, Attention Is All You Need, Arxiv, 2017. |
| [12] | J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Arxiv, 2019. |
| [13] | A. Conneau, A. Baevski, R. Collobert, A. Mohamed and M. Auli, *Unsupervised Cross-lingual Representation Learning for Speech Recognition,* Arxiv, 2020. |
| [14] | R. Ardilla and e. al., *Common Voice: A Massively-Multilingual Speech Corpus,* 2020. |
| [15] | Huggingface, "Datasets Documentation," [Online]. Available: https://huggingface.co/docs/datasets/index. [Accessed 19 Desember 2021]. |
| [16] | A. P. F. Naiborhu and S. N. Endah, "LSTM, Indonesian Continuous Speech Recognition Using CNN and Bidirectional," in *2021 5th International Conference on Informatics and Computational Sciences (ICICoS)*, Semarang, Indonesia, 2021. |
| [17] | B. H. Juang and L. R. Rabiner, "Automatic Speech Recognition – A Brief History of the Technology Development," 10 Agustus 2004. [Online]. Available: https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354\_LALI-ASRHistory-final-10-8.pdf. [Accessed 19 Desember 2021]. |