
7asebatiya 7elwan
arabic-egy-asr

# PHASE1

AUTOMATIC SPEECH RECOGNITION (ASR) SYSTEM

| | | | |
|---|---|---|---|
| Introduction | System Architecture | Methodology | Technical Details |
| Results | Reproducibility | Performance Assessment | |

## 1.introduction:

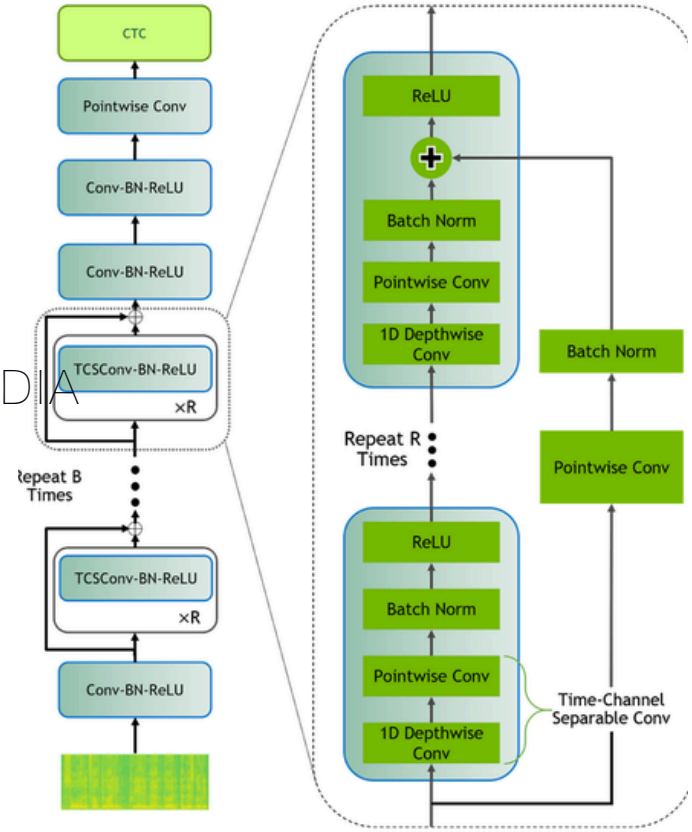
This project implements an Automatic Speech Recognition (ASR) system for Arabic using the QuartzNet architecture. QuartzNet is an end-to-end neural acoustic model that achieves near state-of-the-art accuracy while using fewer parameters than competing models.
The QuartzNet architecture, as described in the original paper: QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions, is composed of multiple blocks with residual connections. Each block consists of one or more modules with 1D time-channel separable convolutional layers, batch normalization, and ReLU layers. The model is trained using Connectionist Temporal Classification (CTC) loss.

# 2.System Architecture:

## QuartzNet15x5 Model Configuration

- **Structure:**
  - Input Layer.
  - 15 QuartzNet Blocks.
  - Output Layer.
- **Each QuartzNet Block Contains:**
  - 1D time-channel separable convolutional layers.
  - Batch normalization.
  - ReLU activation.
  - Residual connections.
- **Details:**
  - Filters: 256 to 512 across different layers.
  - Kernel sizes: 33 to 87.

develop Smaller Speech Recognition Models with the NVIDIA NeMo Framework NVIDIA Technical Blog :



# 3.Methodology :

**Data Preparation:**

- Used the Arabic speech dataset from the MTC-AIC2 organizers.
- Processed data into manifest files for training and validation.

**Model Configuration:**

- Used the NeMo toolkit for configuring and training the QuartzNet model.
- Configuration includes:
  - Audio preprocessing parameters
  - Model architecture details
  - Training hyperparameters

**Training:**

- Trained the model with PyTorch Lightning using:
  - Mixed precision (16-bit)
  - GPU acceleration
  - Maximum of 20 epochs
  - Novograd optimizer with Cosine Annealing learning rate schedule

**Evaluation:**

- Evaluated the model's performance using Word Error Rate (WER) on a validation set.

# 4:Technical Details

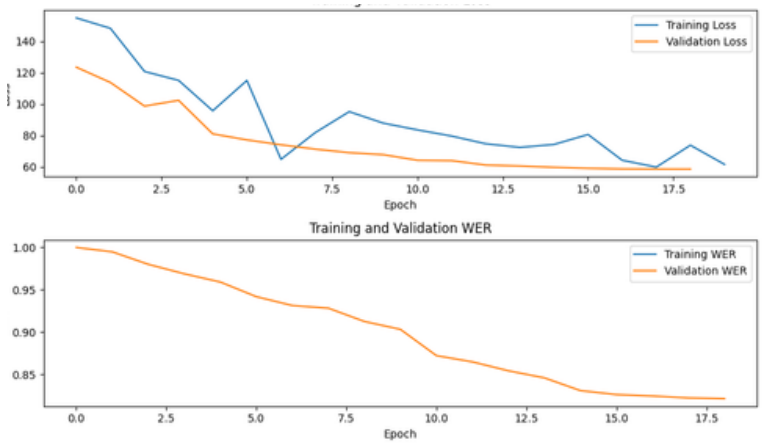
**Model Configuration:**

- Preprocessor: AudioToMelSpectrogramPreprocessor
  - Window size: 0.02
  - Window stride: 0.01
  - Features: 64
  - n_fft: 512
- **Encoder: ConvASREncoder**
  - 15 blocks with varying configurations
  - Separable convolutions
  - Residual connections
- **Decoder: ConvASRDecoder**
  - 1024 input features
  - 41 output classes (Arabic characters)
- **Training Configuration:**
  - Optimizer: Novograd
  - Learning rate: 0.01
  - Betas: [0.8, 0.5]
  - Weight decay: 0.001
- **Learning Rate Schedule:**
  CosineAnnealing
- **Data Augmentation:**
  SpecAugment

# 5.Results

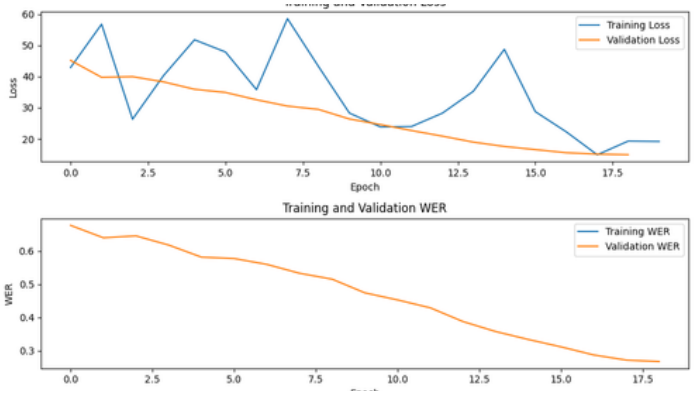
We trained the model for various numbers of epochs (20, 40, 60, 80, 100, 120) and plotted the results:

## 20-epochs:



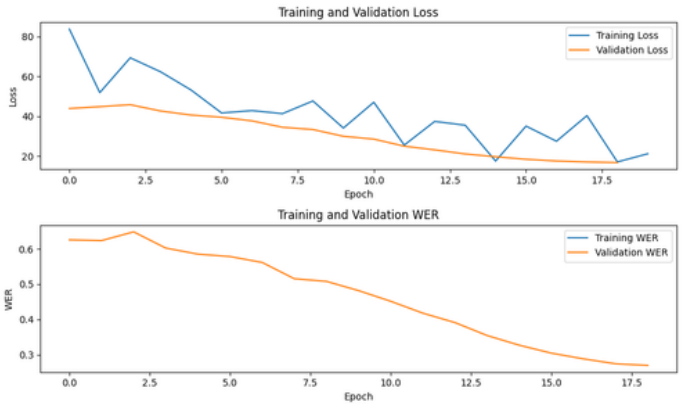

## 40-epochs:


## 60-epochs:




## 80-epochs:

100-epochs:




120-epochs:


# 6.Reproducibility :

**To reproduce our results:**

**1.Install the required dependencies:**

- apt-get update && apt-get install -y libsndfile1 ffmpeg
- pip -q install nemo_toolkit['asr'] Cython packaging
- pip install torch
- pip install pytorch-lightning
- pip install omegaconf
- pip install pandas
- pip install matplotlib

- Prepare  Data:
- Update the manifest file paths in the configuration.
- Create the Manifest File:
- Use the following Python code to create the manifest file. Ensure you have a CSV file prepared with the following structure:

| audio | transcript |
| --- | --- |
| path/to/audio_folder/file1.wav | Transcript of the first audio |
| path/to/audio_folder/file2.wav | Transcript of the second audio |

## 🔗 generate the manifest file

3.Use the provided Python script from training_notebook.ipynb to train the model. Please make sure you have the necessary computational resources (GPU recommended).

4.Adjust hyperparameters as needed for your specific use case.

# 7.Performance Assessment:

**To Assess the Performance of the Model:**

1. Primary Metric:
   - Use the validation set Word Error Rate (WER).
2. Comparison:
   - Compare the model's performance with other state-of-the-art ASR systems for Arabic.
3. Evaluation:
   - Assess the model's inference speed and resource usage.
4. Generalization:
   - Test the model on various Arabic dialects and accents to evaluate its generalization capabilities.
5. Metrics:
   - For a more comprehensive evaluation, consider using additional metrics such as Character Error Rate (CER).

## Conclusion

This document covers the model configuration, training, and evaluation of our QuartzNet ASR system. For more details, see the Manifest_code.