Ses Tanıma Kullanarak Evrişimli Derin Sinir Ağları makale raporu

Giriş
Ayad Alsobhani ve ekibi tarafından gerçekleştirilen "Speech Recognition using Convolutional Deep Neural Networks" başlıklı çalışma, ses tanıma teknolojilerindeki yeniliklere odaklanmaktadır. Bu araştırma, özellikle gürültülü ortamlardan toplanan ses verilerini kullanarak derin öğrenme modellerinin nasıl etkinleştirilebileceğini inceler. Çalışma, derin evrişimli sinir ağlarının (CNN) ses tanıma uygulamalarında nasıl kullanılabileceğini göstermektedir.

Metodoloji:
Araştırmacılar, farklı ortamlardan ve farklı koşullar altında toplanan ses verileri üzerinde çalışmışlardır. Toplam 30 katılımcıdan ses örnekleri toplanmıştır. Bu örnekler, altı farklı kontrol kelimesi içermekte ("başla", "dur", "ileri", "geri", "sağ", "sol") ve bu sesler laboratuvar, sokak, park ve pazar gibi çeşitli yerlerde kaydedilmiştir. CNN modeli, bu verileri işlemek ve kelimeleri doğru bir şekilde sınıflandırmak için kullanılmıştır.

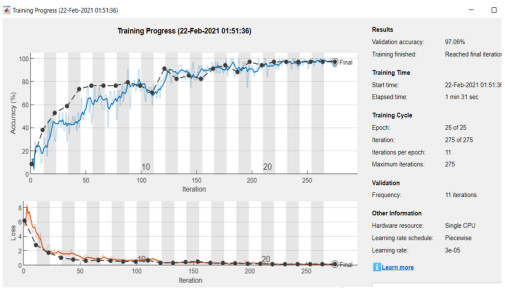
Bulgular:
Model, toplanan veri seti üzerinde %97.06 oranında bir doğrulukla sesleri tanıyabilmiştir. Bu yüksek doğruluk oranı, CNN'in gürültülü ortamlarda dahi etkili bir şekilde ses tanıma yeteneğine sahip olduğunu göstermektedir. Araştırma, CNN mimarisinin, ses verilerini sınıflandırma konusunda üstün performans sergileyebileceğini ve gerçek dünya koşullarında uygulanabilirliğini kanıtlamaktadır.

Sonuç:
Alsobhani ve ekibinin çalışması, ses tanıma sistemlerinin geliştirilmesinde CNN kullanımının potansiyel faydalarını ortaya koymaktadır. Araştırma, özellikle çeşitli ve gürültülü veri kümeleri üzerinde yüksek doğruluk oranları elde edilmesi açısından önemli bir referans noktası sağlar. CNN, gelişmiş öğrenme kabiliyetleri ve adaptasyon yeteneği ile ses tanıma teknolojilerinde devrim yaratma potansiyeline sahiptir.

Bu çalışma, ses tanıma teknolojilerinin daha ileri taşınması için sağlam bir temel oluşturmuş ve ses tanıma sistemlerinin gelecekteki uygulamaları için yeni kapılar açmıştır. Araştırmanın sonuçları, ses tanıma teknolojilerinin evrimsel gelişiminde önemli bir adım olarak değerlendirilmekte ve bu teknolojilerin geniş bir uygulama alanına adaptasyonu için kapsamlı bir yol haritası sunmaktadır.

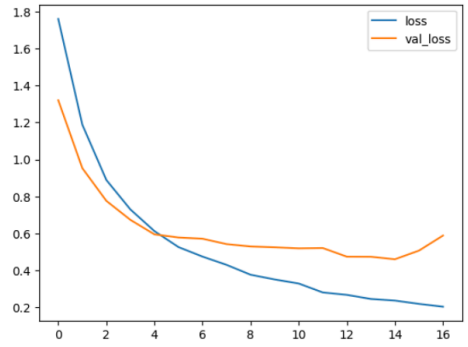
Benim bulgularım ile karşılaştırma

Makalede modeli 275 iterasyon ile eğitmişler ben kendimde 150 iterasyon ile eğittim benim



(Makalede olan bulgular)

burdan anladığımız üzere modelleri güzel uyum sağlamış durumda



Benim eğitimimde pek uyumlu bir sonuç çıkmadığı gözüküyor
Bunun sebepleri arasında benim verilerimin uymsuz olması ve modelimin yeterli olmaması olabilir

Projeyi tekrarlarken yaptığım adımlar

Öncelik ile veristeini indirdim

```
DATASET_PATH = 'data/mini_speech_commands'

data_dir = pathlib.Path(DATASET_PATH)
if not data_dir.exists():
  tf.keras.utils.get_file(
      'mini_speech_commands.zip',
      origin="http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands.zip",
      extract=True,
      cache_dir='.', cache_subdir='data')
```

daha sonra veri setini ayırdım

```
train_files = filenames[:6400]
val_files = filenames[6400: 6400 + 800]
test_files = filenames[-800:]

print('Training set size', len(train_files))
print('Validation set size', len(val_files))
print('Test set size', len(test_files))
```

```
Training set size 6400
Validation set size 800
Test set size 800
```

ileride içlerinden label gibi değerlerini öprenmek için fonksiyonları ekledim

```
def decode_audio(audio_binary):
  # Decode WAV-encoded audio files to `float32` tensors, normalized
  # to the [-1.0, 1.0] range. Return `float32` audio and a sample rate.
  audio, _ = tf.audio.decode_wav(contents=audio_binary)
  # Since all the data is single channel (mono), drop the `channels`
  # axis from the array.
  return tf.squeeze(audio, axis=-1)
```

```
def get_label(file_path):
  parts = tf.strings.split(
      input=file_path,
      sep=os.path.sep)
  # Note: You'll use indexing here instead of tuple unpacking to enable this
  # to work in a TensorFlow graph.
  return parts[-2]
```

```
def get_waveform_and_label(file_path):
  label = get_label(file_path)
  audio_binary = tf.io.read_file(file_path)
  waveform = decode_audio(audio_binary)
  return waveform, label
```

verinin ön işlemelerini yaptım

```
def preprocess_dataset(files):
  files_ds = tf.data.Dataset.from_tensor_slices(files)
  output_ds = files_ds.map(
      map_func=get_waveform_and_label,
      num_parallel_calls=AUTOTUNE)
  output_ds = output_ds.map(
      map_func=get_spectrogram_and_label_id,
      num_parallel_calls=AUTOTUNE)
  return output_ds
```

```
train_ds = spectrogram_ds
val_ds = preprocess_dataset(val_files)
test_ds = preprocess_dataset(test_files)
```

```
batch_size = 64
train_ds = train_ds.batch(batch_size)
val_ds = val_ds.batch(batch_size)
```

```
train_ds = train_ds.cache().prefetch(AUTOTUNE)
val_ds = val_ds.cache().prefetch(AUTOTUNE)
```

ve modeli oluşturup eğittim

Input shape: (124, 129, 1)

Model: "sequential_1"

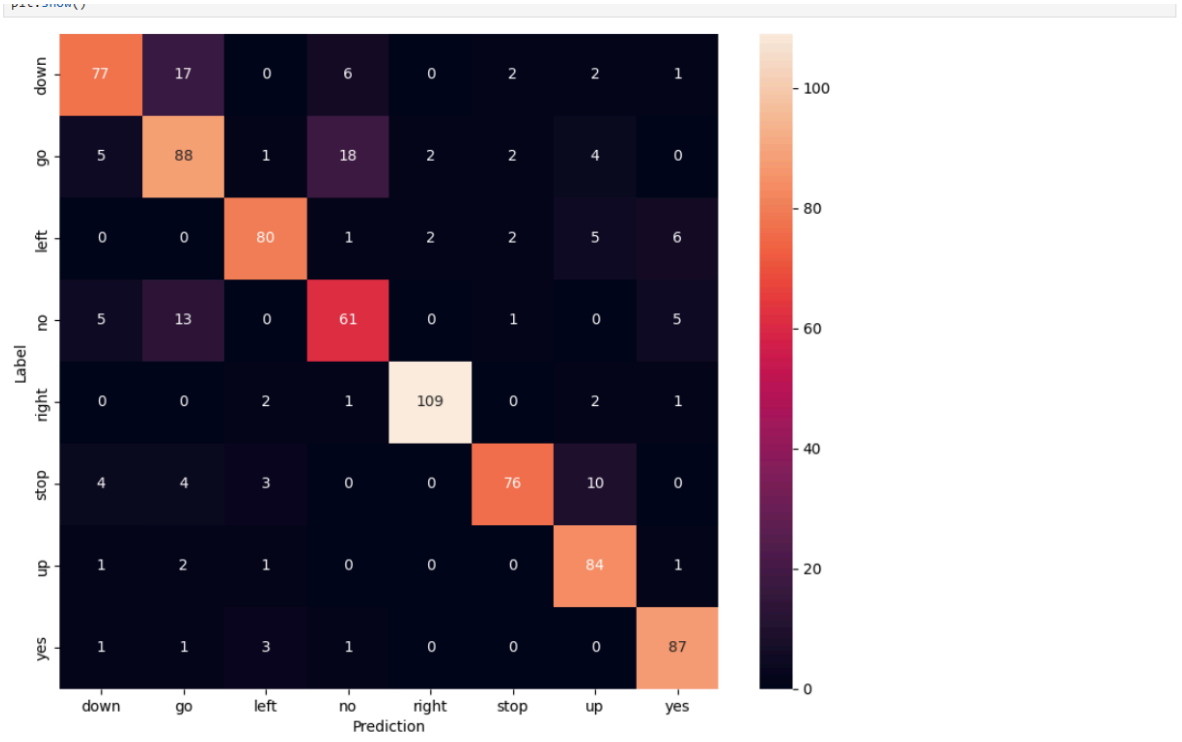
Layer (type)	Output Shape	Param #
resizing_1 (Resizing)	(None, 32, 32, 1)	0
normalization_1 (Normalization)	(None, 32, 32, 1)	3
conv2d_2 (Conv2D)	(None, 30, 30, 32)	320
conv2d_3 (Conv2D)	(None, 28, 28, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_2 (Dense)	(None, 128)	1,605,760
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 8)	1,032

Total params: 1,625,611 (6.20 MB)

Trainable params: 1,625,608 (6.20 MB)

Non-trainable params: 3 (16.00 B)

daha sonra confusion matrix ini oluşturdum



ve tahminleri test edip modeli kaydettim .