BAB II

TINJAUAN PUSTAKA

2.1. Landasan Teori

Pada penilitian ini, sebuah aplikasi *speech recognition* kamus geografi dirancang menggunakan metode *Convolutional Neural Network* berbasis aplikasi komputer. Penelitian terdahulu adalah penelitian oleh Edbert Hansel [2] dengan membuat aplikasi *speech recognition* untuk kamus geografi berbahasa Indonesia. Abdel-Hamid [1] berhasil membuktikan bahwa CNN mempunyai yang sangat baik dalam mengenali suara yang dapat diterapkan ke aplikasi *speech recognition.*

Analisa metode CNN ini dibuat pada sistem berbasis aplikasi *desktop*. Input adalah hasil rekaman yang diucapkan oleh user. Proses pelatihan kata menggunakan metode *Convolutional Neural Network* (CNN) yang diterapkan pada framework TensorFlow. Rancangan model CNN dalam aplikasi dibuat berdasarkan penelitian yang dilakukan oleh Tara N. Sainath dan Carolina Parada [3] dari Google pada konferensi INTERSPEECH 2015. Kelebihan aplikasi ini adalah lebih akurat dalam fitur *speech recognition*, sehingga hasil yang dikeluarkan akurat dan anak-anak Indonesia lebih tertarik untuk mempelajari Geografi.

2.2. TensorFlow

TensorFlow merupakan *library open source* untuk perhitungan numerik menggunakan *data flow graphs*. *Nodes* pada graph mewakili operasi matematika, sedangkan *graph edges* mewakili array multidimensi (tensor) yang mengalir dari keduanya. TensorFlow awalnya dikembangkan oleh *researchers* dan *engineers* dari

Google Brain di dalam Machine Intelligence Research Google, dengan tujuan melakukan riset *machine learning* dan *deep neural networks* [4].

TensorFlow mendukung pengembangan aplikasi dengan frameworknya di sistem operasi Windows dan Linux. Semua proses di dalam TensorFlow melibatkan tensor. Tensor adalah generalisasi dari vector atau matriks untuk mencapai dimensi yang lebih tinggi. Secara internal, TensorFlow membuat tensor sebagai array n-dimensi yang berisi tipe data dasar. Semua elemen di dalam Tensor mempunyai tipe data yang sama dan tipe datanya harus diketahui [4].

TensorFlow dapat digunakan untuk melakukan algoritma *machine learning* dan dapat mengimplementasikan algoritma tersebut [5]. Perhitungan di dalam TensorFlow dapat dieksekusi dalam sistem yang sangat luas, dari perangkat *mobile* seperti handphone hingga komputer canggih dengan perhitungan kartu GPU. Karena sistem yang fleksibel, banyak algoritma yang rumit dapat ditulis ke dalam TensorFlow, termasuk pelatihan dari algoritma *Deep Neural Network.*

2.2.1. Tensor

Tensor adalah bagian inti dari TensorFlow. Tensor dapat mempunyai *rank* untuk tiap dimensi yang dipunyai. Biasanya disebut *order* atau *degree*. Rank tersebut dapat dilihat pada tabel 2.1.

Tabel 2.1 Rank TensorFlow *[4]*

Rank	Entitas Matematika
0	Scalar (hanya besaran)
1	Vector (besaran dan arah)
2	Matriks (tabel dari angka)
3	3-Tensor (kubus dari angka)
N	n-Tensor

Tensor dapat diisi berbagai macam tipe data. *Signed* atau *unsigned* integer dari 8 bits sampai 64 bits. Float dan double IEEE, angka kompleks, string juga dapat dimasukkan.

2.2.2. Variable

Variable di dalam TensorFlow adalah tempat untuk menyimpan *persistent and shared state* variabel yang nantinya dimanipulasi oleh program. Cara terbaik membuat variabel adalah dengan menggunakan fungsi tf.get_variable(). Fungsi ini digunakan untuk mengakses sebuah variable dari banyak replika. Selain itu dengan variabel, variabel yang sudah pernah dibuat untuk dapat digunakan kembali [4].

2.2.3. Operasi dan Kernel

Operasi di dalam TensorFlow mempunyai nama dan mewakili komputasi yang abstrak seperti penambahan atau perkalian matriks. Operasi dapat mempunyai atribut dan semua atribut harus diberikan saat pembangunan *graph* agar dapat membuat *node* melakukan sebuah operasi [5]. Kegunaan atribut yang paling umum

adalah untuk membuat operasi antar 2 tipe variabel yang berbeda. Contohnya adalah penambahan 2 tensor bertipe float ke dalam 2 tensor bertipe int32.

Kernel adalah implementasi operasi di dalam *device* tertentu. Bisa dijalankan dalam CPU ataupun GPU. TensorFlow *binary* dapat mendefinisikan kumpulan operasi dan kernel yang tersedia melalui mekanisme registrasi. Sehingga jika diperlukan operasi atau kernel dari luar, dapat ditambahkan via registrasi.

2.2.4. Sessions

Klien berinteraksi dengan sistem TensorFlow melalui pembuatan *Session*. Untuk membuat *computation graph*, antar muka *Session* mendukung *method Extend* untuk menggabungkan *Session* yang sedang dijalankan dengan tambahan *nodes* atau *edges.* Operasi utama lain yang didukung oleh *Session* adalah Run, dengan mengambil kumpulan nama keluaran yang harus dihitiung dan kumpulan tensor yang harus diberikan ke dalam *graph* untuk menghasilkan beberapa keluaran *nodes*. Dengan Run, TensorFlow dapat menjalankan semua perhitungan *node*s dengan tujuan mendapatkan keluaran yang diminta [5].

2.3. Speech Recognition

Automatic Speech Recognition oleh mesin sudah menjadi tujuan riset lebih dari empat dekade dan telah menjadi inspirasi untuk banyak film *sci-fi* seperti komputer HAL di film “A Space Odyssey” dan robot R2D2 di film “Star Wars”. Tetapi untuk mencapai tujuan tersebut, para ilmuan masih jauh dari memperoleh mesin yang dapat mengenali setiap subjek dari semua orang di dalam segala kondisi [6]. Teknologi ini memungkinkan sebuah mesin untuk dapat mengolah suara dari

seseorang dan memahami kata yang diucapkan. Cara kerjanya adalah dengan mengubah suara orang menjadi sinyal digital yang berisi angka, kemudian angka tersebut disesuaikan dengan pola yang sudah dipelajari untuk menentukkan kata yang diucapkan oleh seseorang. Nantinya mesin juga dapat menampilkan keluaran berupa bentuk teks ke user.

Salah satu aspek tersulit untuk melakukan riset dalam *speech recognition* oleh mesin adalah karena diperlukan lebih dari satu cabang ilmu dan langkah yang biasanya dilakukan oleh periset adalah menerapkan pendekatan monolitik ke masalah individu. Beberapa disiplin ilmu yang sudah pernah diterapkan ke permasalahan *speech recognition* [6]:

- Pengolahan Sinyal: Pengolahan Sinyal adalah proses ekstraksi informasi yang relevan dari sinyal suara. Di dalam pengolahan sinyal, *spectral analysis* termasuk salah satu contoh penerapan untuk mengetahui properti *time-varying* dari sinyal suara. Beberapa tipe *preprocessing* sinyal juga termasuk dalam membuat sinyal suara bagus untuk diolah nantinya.
- *Physics* (*acoustics*): *Physics* adalah ilmu memahami hubungan antara sinyal suara fisik dan mekanisme fisiologi (mekanisme sistem vokal manusia) yang menghasilkan suara yang dapat didengar (mekanisme sistem pendengaran manusia).
- *Pattern Recognition*: *Pattern Recognition* adalah kumpulan algoritma yang digunakan untuk mengelompokkan data sesuai

dengan polanya (*clustering)* dan mencocokkan pasangan pola untuk dibandingkan fitur polanya.

- Teori Komunikasi dan Informasi: Teori ini adalah prosedur untuk mengestimasi parameter dari model statistik. Metode untuk mendeteksi keberadaan pola suara tertentu. Ini juga merupakan algoritma *coding* dan *decoding*. Digunakan untuk mencari jalan terbaik dari *grid* yang jumlahnya terbatas untuk menentukkan urutan kata terbaik yang dikenali.
- *Linguistics*: *Linguistics* adalah hubungan antar suara (*phonology*), kata di dalam bahasa (*syntax*), arti dari kata yang diucapkan (*semantics*), dan *sense* yang didapatkan dari arti (*pragmatics*). Di dalam ilmu ini adalah metodologi dari *grammar* dan pengolahan kata.
- Fisiologi: Fisiologi adalah pemahaman dari mekanisme tingkat tinggi dari sistem saraf manusia untuk menghasilkan suara dan menghasilkan persepi di manusia. Banyak teknik modern yang mencoba menerapkan pengetahuan ini di *framework artificial neural networks*.
- Ilmu Komputer: Dampak dari Ilmu Komputer adalah terhadap pembelajaran algoritma yang efisien untuk diimplementasikan, di *software* maupun *hardware*, juga bermacam metode yang digunakan untuk praktek sistem pengenal suara.

- Psikologi: Ilmu yang mempelajari faktor yang memungkinkan teknologi untuk digunakan oleh manusia dalam kehidupan sehari-hari.

Dalam pendekatan *Automatic Speech Recognition* oleh mesin, ada 3 jenis pendekatan[6]:

- Pendekatan *acoustic-phonetic*
- Pendekatan *pattern recognition*
- Pendekatan *artificial intelligence*

Pendekatan melalui *acoustic-phonetic* sudah dipelajari lama dan lebih dari 40 tahun. Tetapi karena beberapa alasan, pendekatan ini belum mencapai kesuksesan dibandingkan pendekatan alternatif lainnya. Pendekatan ini berpusat pada teori akustik fonetik yang membedakan tiap unit fonetik dalam sebuah bahasa dan unit fonetik ini mempunyai karakteristik dalam sinyal suaranya, atau dalam spectrumnya. Fase pertamanya adalah segmentasi dan *labeling* dimana fase ini memecah sinyal suara ke bentuk *discrete*, dimana properti akustik dari sebuah sinyal mewakili sebuah unit fonetik, dan memberi label fonetik ke bagian yang dipecah sesuai properti akustiknya. Fase kedua adalah menebak kata dari kumpulan unit fonetik yang didapat di fase satu[6].

Pendekatan melalui *pattern-recognition* mempunyai dua langkah, yaitu pelatihan dari pola suara dan memahami pola melalui perbandingan pola. Pemahaman akan suara diberikan ke sistem melalui prosedur pelatihan. Algoritma dibuat dengan *training set* yang disediakan. Pelatihan ini seharusnya bisa

mengenali properti akustik dari sebuah pola. Mesin dapat melakukan *pattern classification* karena mesin mempelajari properti akustik mana yang dapat diandalkan selama dalam proses pelatihan. Pendekatan ini adalah pendekatan yang cocok untuk pengenalan suara karena tiga alasan[6]:

- Pemakaian yang mudah
- Ketahanan dan ketetapan untuk kamus, pengguna, set fitur, algoritma pembanding, dan aturan pengambilan keputusan yang berbeda.
- Terbukti sangat bagus

Sedangkan pendekatan *artificial intelligence* adalah pendekatan yang menggabungkan pendekatan *acoustic-phonetic* dan pendekatan *pattern recognition*. Pendekatan ini mengeksploitasi konsep kedua pendekatan lainnya. Pendekatan ini mencoba menjalankan prosedur pengenalan seperti seseorang menerapkan kepintarannya dalam memvisualisasi, menganalisa, dan akhirnya membuat keputusan berdasarkan fitur akustik. Sehingga teknik yang digunakan dalam metode ini meliputi penggunaan sistem ahli dalam segmentasi dan labeling, sehingga langkah paling krusial dan tersulit dapat dilakukan tidak hanya dengan informasi akustik di pendekatan *acoustic-phonetic* saja. Tetapi pendekatan ini juga dapat belajar dan beradaptasi seiring waktu dan menggunakan *neural networks* untuk mempelajari hubungan antar fonetik, *input*, dan diskrimasi antar kelas suara yang mirip[6].

Penggunaan *neural networks* dapat dikatakan sebagai implementasi arsitektur yang dapat diterapkan dalam tiga pendekatan klasik. Konsep dan ide menerapkan *neural networks* ke permasalahan pengenalan suara masih baru-baru ini[6]. Sehingga *neural networks* dianggap sebagai kunci menyelesaikan masalah pengenalan suara di beberapa kasus nyata. Penlitian ini nantinya juga berpusat pada penggunaan *neural networks* sebagai dasar arsitektur.

2.4. Convolutional Neural Network

Convolutional Neural Network (CNN) menyediakan *shift invariance* terhadap waktu dan cakupan dan penting untuk mencapai performa *state-of-the-art* untuk image recognition[7]. Tetapi CNN juga sudah terbukti dapat meningkatkan akurasi pengenalan suara disbanding *Deep Neural Network* murni di beberapa tugas tertentu [1], [8]. Untuk mendukung CNN, beberapa titik komputasi perlu diimplementasikan.

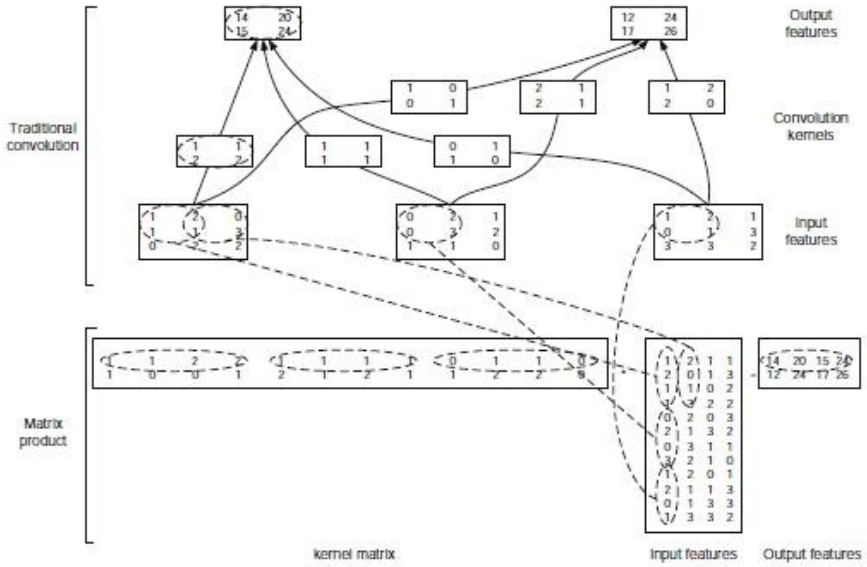
Di penelitian sebelumnya penggabungan *deep neural network – hidden Markov Model* (DNN-HMM) menunjukkan perkembangan pada performa pengenalan suara dibandingkan model konvensional *Gaussian mixture model* (GMM-HMM). Penambahan pada performa adalah karena kapasitas DNN untuk memodelkan korelasi kompleks pada fitur suara. Terlebih lagi, CNN dapat menurunkan *error rate reduction*[1].

Convolution dalam kasus ini adalah proses konvolusi produk berelemen dari sebuah kernel menjadi sebuah gambar. *Input* dari *convolution node* mempunyai 3 kanal (direpresentasikan sebagai matriks 3x3) dan keluarannya mempunyai 2 kanal

(direpresentasikan sebagai matriks 2x2). Kanal adalah pandangan dari gambar yang sama. Contohnya sebuah gambar RGB bisa direpresentasikan menjadi tiga kanal: R, G, dan B, setiap kanal ukurannya sama[7].

Dalam setiap pasangan kanal *input* dan *output* ada kernelnya. Jumlah kernel sama dengan jumlah produk dari kanal *input Cx dan jumlah kanal* output *Cv*. Pada Gambar 2.1 di bawah, *Cx* = 3, *Cv* = 2 dan jumlah kernelnya adalah 6. Setiap kernel **K***kℓ* dari kanal input *k* dank anal output *ℓ* adalah matriks. Kernel bergerak mengikuti mengikuti *input* dengan *subsampling rate Sr* dan *Sc* berurutan pada arah vertikal dan horizontal. Untuk setiap *output* kanal *ℓ* dan irisan *input* (i,j), dimana **Y***kij* mempunyai ukuran yang sama dengan **K***kℓ*.

$$\boldsymbol{v}_{\ell ij}\,(\boldsymbol{K},\boldsymbol{Y}) = \sum_{\boldsymbol{k}} vec\,(\boldsymbol{K}_{\boldsymbol{k}\ell}) \circ vec\,(\boldsymbol{Y}_{\boldsymbol{k}ij}) \qquad (2.1)$$



Gambar 2.1 Contoh Operasi Konvolusi*[7]*

Fungsi evaluasi ini melibatkan banyak operasi matriks keccil dan bisa lambat. Chellapila et al. [9] mengajukan teknik untuk mengkonversi semua operasi matriks kecil menjadi sebuah matriks produk besar yang terlihat di bawah Gambar 2.1. Dengan ini, semua parameter kernel dapat digabungkan menjadi sebuah kernel matriks besar **W** seperti yang terlihat di bawah kiri Gambar 2.1. Agar titik konvolusi dapat digunakan oleh titik konvolusi yang lain, konversi harus dibuat sedikit berbeda dari yang dirancang oleh Chellapila et al. [9] dengan melakukan *transpose* kedua kernel matriks dan fitur input matriks serta orde matriks di produk. Dengan perubahan ini, setiap sampel di keluaran dapat direpresentasikan dengan kolom *Or* × *Oc dari Cv* × 1 vector yang dapat dirubah bentuknya menjadi sebuah kolom, dimana

$$O_r = \begin{cases} \frac{I_r - K_r}{S_r} + 1 & \text{no padding} \\ \frac{(I_r - \text{mod}(K_r, 2))}{S_r} + 1 & \text{zero padding} \end{cases} \quad (2.2)$$

adalah jumlah baris dari *output* gambar, dan

$$O_c = \begin{cases} \frac{I_c - K_c}{S_c} + 1 & \text{no padding} \\ \frac{(I_c - \text{mod}(K_c, 2))}{S_c} + 1 & \text{zero padding} \end{cases} \quad (2.3)$$

adalah jumlah baris dari *output* gambar dimana *Ir* and *Ic* adalah, berurutan, jumlah baris dan kolom dari input gambar. *Kr* and *Kc* adalah, berurutan, jumlah baris dan kolom di masing-masing kernel. Gabungan kernel matriksnya berukuran *Cv* × *(Or*

× *Oc* × *Cx)*, dan fitur *input* matriks yang menumpuk berukuran *(Or* × *Oc* × *Cx)* × *(Kr* × *Kc)*[7].

Dengan konversi ini, perhitungan dari titik konvolusi dengan operan **W**,**Y** menjadi[7]:

$$\boldsymbol{X}(\boldsymbol{Y}) \leftarrow Pack\ (\boldsymbol{Y}) \tag{2.4}$$

$$V\ (\boldsymbol{W},\boldsymbol{Y}) \leftarrow \boldsymbol{WX} \tag{2.5}$$

$$\nabla_W^J \leftarrow \nabla_W^J + \nabla_n^J X^T \tag{2.6}$$

$$\nabla_X^J \leftarrow W^T \nabla_n^J \tag{2.7}$$

$$\nabla_Y^J \leftarrow \nabla_Y^J + Unpack\ (\nabla_X^J) \tag{2.8}$$

Teknik ini memungkinkan pengerjaan paralel untuk operasi matriks yang besar, tetapi menambah pekerjaan di membungkus dan membuka matriks (*pack* dan *unpack*). Dalam kebanyakan kondisi, lebih banyak untungnya daripada kerugiannya. Saat membuat titik konvolusi dengan titik tambahan dan fungsi *element-wise nonlinear*, bias dan *nonlinearity* bisa ditambahkan ke operasi konvolusi[7].

Ada 2 jenis pooling yang biasanya digunakan[7]:

- *MaxPooling*: Menerapkan operasi *pooling* maksimum ke *input* X didalam jendela berukuran *Kr* × *Kc* untuk setiap kanal. Operasi jendela bergerak mengikuti *subsampling rate* dari *input Sr* dan *Sc*

pada arah vertikal dan horizontal secara berurutan. Operasi *pooling* tidak merubah jumlah kanal, sehingga *Cv = Cx*.

- *AveragePooling:* Menerapkan rata-rata operasi *pooling* ke *input* X pada jendela berukuran *Kr* × *Kc* untuk setiap kanal. Operasi jendela bergerak mengikuti *subsampling rate* dari *input Sr* dan *Sc* pada arah vertikal dan horizontal secara berurutan.

Arsitektur dari sebuah CNN berisi sebuah pasang lapisan, lapisan konvolusi dan lapisan *pooling*. *Input* berisi angka dari fitur yang dilokalisasi sebagai peta fitur. Ukuran dari peta fitur menjadi kecil pada lapisan atas, seiring dengan semakin banyaknya operasi konvolusi dan *pooling* dilakukan. Biasanya satu atau lebih *hidden layers* yang terhubung ditambahkan di atas lapisan CNN untuk menggabungkan fitur dari semua *frequency band*s sebelum dimasukkan ke *output layer*[1].

2.5. Kerangka Berpikir

Penelitian ini dilakukan dengan proses mengumpulkan data terlebih dahulu, lalu melakukan *training* dari data yang sudah dikumpulkan dengan metode *Convolutional Neural Network* yang berada pada *framework* TensorFlow. Lalu membangun aplikasi *speech recognition* dengan menggunakan komputer sebagai perangkat keras untuk menerima input berupa kata yang diucapkan oleh pengguna. Setelah itu melakukan *testing* dan menguji akurasi dari aplikasi yang sudah dibuat.