PENGGUNAAN MODEL TRANSFORMER PADA *AUDIOVISUAL SPEECH RECOGNITION* UNTUK BAHASA INDONESIA

TESIS

Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Magister dari
Institut Teknologi Bandung

Oleh

GUGY LUCKY KHAMDANI
NIM: 23517041
(Program Studi Magister Informatika)

PROGRAM STUDI MAGISTER INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

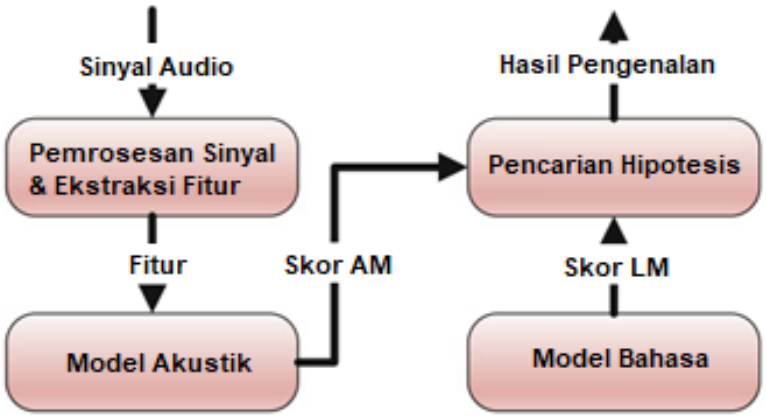
INSTITUT TEKNOLOGI BANDUNG

Februari 2019

Bab II Tinjauan Pustaka

II.1 Sistem Pengenalan Ucapan Otomatis

Sistem pengenalan ucapan otomatis atau disebut juga *automatic speech recognition* (ASR) adalah proses pengenalan atau penerjemahan bahasa lisan dalam bentuk sinyal audio menjadi teks secara otomatis oleh komputer. Pada umumnya, arsitektur dari sebuah system ASR bisa dibagi menjadi empat buah komponen utama, yaitu pemrosesan sinyal dan ekstraksi fitur, model akustik atau acoustic model (AM), model bahasa atau language model (LM), dan pencarian hipotesis (Yu dan Deng, 2014).



Gambar II.1: Arsitektur umum dari sistem ASR (Yu dan Deng, 2014)

II.1.1 Pemrosesan Sinyal dan Ekstraksi Fitur

Komponen pemrosesan sinyal dan ekstraksi fitur mengambil masukan berupa sinyal audio mentah dan diproses agar *noise*-nya dihilangkan, mengubah sinyal dari domain waktu ke domain frekuensi, dan mengekstraksi fitur-fitur yang paling penting dari sinyal tersebut sehingga cocok dengan komponen model akustik.

II.1.2 Model Akustik

Komponen model akustik tersebut mengintegrasikan pengetahuan mengenai akustik dan fonetik, dan dengan menggunakan fitur yang diekstraksi di komponen sebelumnya lalu menghasilkan skor AM untuk fitur tersebut.

Riset Han dkk. (2018) menunjukkan bahwa model akustik dengan *word error rate*

(WER) terbaik untuk korpus bahasa Inggris yaitu sebesar 5.0% pada dataset Switchboard dan 9.1% pada dataset CallHome, didapatkan dengan menggunakan dense TDNN-LSTM. Fitur yang digunakan adalah MFCC dengan dimensi sebesar 39, sama seperti yang dilakukan oleh Xiong dkk. (2017) akan tetapi model akustik yang digunakan adalah CNN-BLSTM.

Selain itu, terdapat riset lain yang menggunakan pendekatan *sequence-to-sequence*, seperti Chan dkk. (2015) yang menggunakan *Bidirectional* LSTM (BLSTM) dengan struktur piramid untuk mendapatkan representasi tingkat tinggi dari sinyal audio. Struktur piramid digunakan untuk mengurangi *time-step* yang dibutuhkan untuk mendapatkan representasi dari keseluruhan sinyal audio, yang bisa mencapai ribuan pada sistem ASR. BLSTM struktur piramid tersebut pada pendekatan *sequence-to-sequence* disebut sebagai *encoder*, menghasilkan context vectors yang kemudian digunakan pada *decoder* untuk menghasilkan rangkaian kata-kata sebagai hasil pengenalan. Model ini berhasil mencapai WER 14.1% tanpa menggunakan kamus dan model bahasa, dan berhasil mencapai WER 10.3% jika menggunakan model bahasa. Riset lain yang menggunakan pendekatan *sequence-to-sequence* pada data bahasa Inggris adalah riset Chung dkk. (2017), yang sama seperti riset sebelumnya hanya saja pada *decoder*-nya tidak menerima masukan berupa sinyal audio mentah, tetapi sudah diekstraksi fitur terlebih dahulu menjadi fitur MFCC berdimensi 13 dengan urutan waktu terbalik. Model ini berhasil mencapai WER 17.7% pada dataset LRS.

Riset mengenai ASR pada bahasa Indonesia salah satunya adalah oleh Yuwan (2018), yang menggunakan model DNN-HMM dan dibandingkan dengan model tolak ukur GMM-HMM. Pengujian model dilakukan pada skema tertutup dan terbuka, dan berhasil mencapai penurunan WER dari ASR berbasis GMM-HMM ke ASR berbasis DNN-HMM sebesar 2,53% pada skema tertutup dan 3,89% pada skema terbuka.

II.1.3 Model Bahasa

Komponen model bahasa mengestimasi probabilitas dari rangkaian kata-kata hipotesis dengan cara mempelajari korelasi antar kata-kata tersebut berdasarkan korpus

data latih. Estimasi probabilitas tersebut disebut sebagai skor LM.

II.1.4 Pencarian Hipotesis

Dengan menggunakan skor AM dan LM yang didapatkan dari vektor fitur dan kata-kata kandidat yang mungkin, komponen pencarian hipotesis akan menghasilkan keluaran berupa rangkaian kata-kata dengan skor AM dan LM tertinggi sebagai hasil pengenalan.

II.2 Pemodelan *Sequence*

Pemodelan *sequence* dengan *deep learning* berkembang dengan pesat, terutama menggunakan *recurrent neural network* (RNN), yang telah menunjukkan hasil yang baik pada banyak permasalahan pembelajaran mesin, khususnya untuk permasalahan yang memiliki masukan dan/atau keluaran yang memiliki panjang yang bervariasi. Riset Sutskever dkk. (2014) dan Bahdanau dkk. (2015) menunjukkan bahwa RNN dapat melakukan pemodelan *sequence* dengan baik sama seperti sistem-sistem yang sudah ada untuk permasalahan sulit seperti mesin translasi.

RNN merupakan perluasan dari jaringan saraf tiruan *feedforward* konvensional yang dapat menangani masukan dengan panjang bervariasi. RNN menanganinya dengan menggunakan *recurrent hidden* state yang pengaktifannya bergantung pada *hidden state* pada waktu yang sebelumnya. Secara formal, RNN memperbarui *recurrent hidden state* ht-nya sebagai berikut (Chung dkk., 2014):

$$h_t = \begin{cases} 0 & t = 0 \\ \varphi(h_{t-1} - x_t) & otherwise \end{cases}$$

yang pada hal ini x adalah rangkaian $x = (x_1, x_2 \ldots, x_T)$ dan φ merupakan fungsi non-linier seperti fungsi *logistic sigmoid*. RNN juga bisa memiliki keluaran $y = (y_1, y_2 \ldots, y_T)$ yang juga memiliki panjang yang bervariasi.

Bobot atau parameter yang terdapat pada RNN secara tradisional diperbarui sebagai berikut (Chung dkk., 2014):

$$h_t = g(Wx_t + Uh_{t-1})$$

yang dalam hal ini g merupakan fungsi seperti *logistic sigmoid* atau *hyperbolic tangent*.

Akan tetapi, pelatihan RNN sulit dilakukan untuk menangkap dependensi yang jaraknya jauh karena nilai gradien pada RNN cenderung menghilang atau meningkat drastis. Hal ini membuat optimasi berbasis gradien menjadi sulit untuk dilakukan. Pendekatan yang bisa dilakukan untuk menangani masalah ini bisa dengan melakukan optimasi yang lain yang tidak berbasis gradien, seperti menggunakan *stocahstic gradient desecent*. Pendekatan lainnya yang bisa dilakukan adalah menggunakan arsitektur RNN yang lain yang menggunakan fungsi aktivasi yang lebih mutakhir, seperti *long short-term memory* (LSTM) (Hochreiter dan Urgen Schmidhuber, 1997) dan *gated recurrent unit* (GRU) (Cho dkk., 2014b).

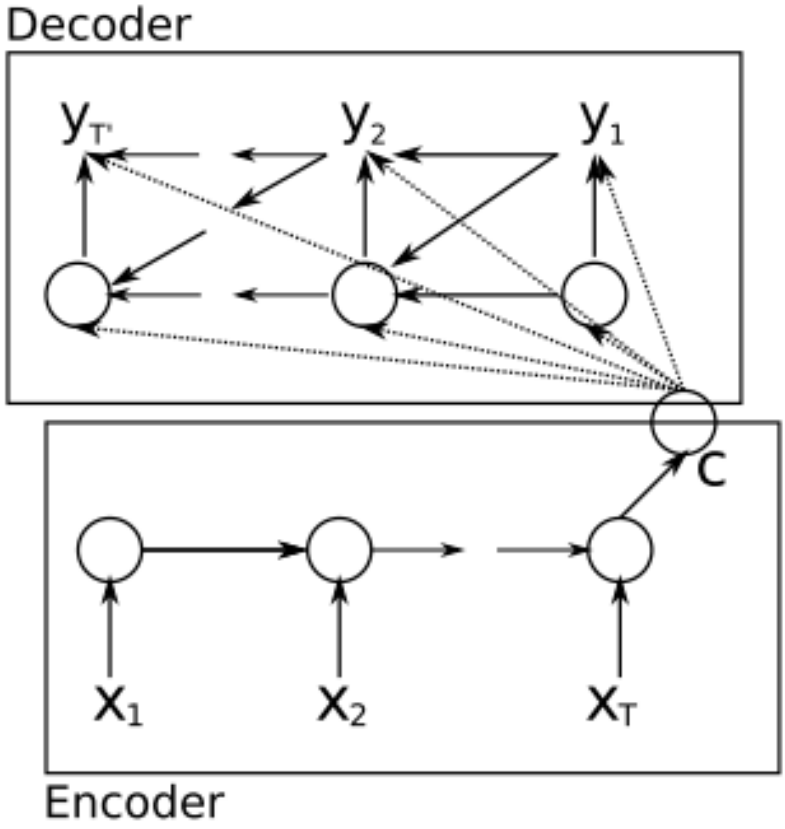
Pemodelan *sequence* yang paling banyak digunakan pada riset-riset terakhir ini adalah menggunakan model *sequence-to-sequence* yang memetakan sebuah rangkaian masukan dengan panjang n menjadi rangkaian keluaran dengan panjang m, dipopulerkan dalam penggunaanya untuk permasalahan terjemahan mesin.

Terdapat dua pendekatan umum untuk model *sequence-to-sequence*, yaitu model *encoder-decoder* berbasis rekurens dan model *encoder-decoder* berbasis *attention*.

II.2.1 Model *Encoder-Decoder* berbasis rekurens

Model dasar *sequence-to-sequence* diperkenalkan oleh Cho dkk. (2014a), yang terdiri dari dua buah *recurrent neural network* (RNN), yaitu *encoder* yang bertugas untuk merepresentasikan rangkaian masukan menjadi sebuah *fixed-length vector*, dan *decoder* yang bertugas untuk menghasilkan rangkaian keluaran berdasarkan vektor yang didapatkan dari encoder tadi.

Selain itu ada riset dari Sutskever dkk. (2014) yang sama menggunakan arsitektur *encoder-decoder* akan tetapi berbeda dengan arsitektur sebelumnya yang hanya mengintegrasikan jaringan saraf tiruannya ke dalam sistem *statistical machine translation* (SMT), model ini merupakan model yang sepenuhnya menggunakan RNN dan proses pelatihannya dilakukan secara *end-to-end*. Hanya saja model

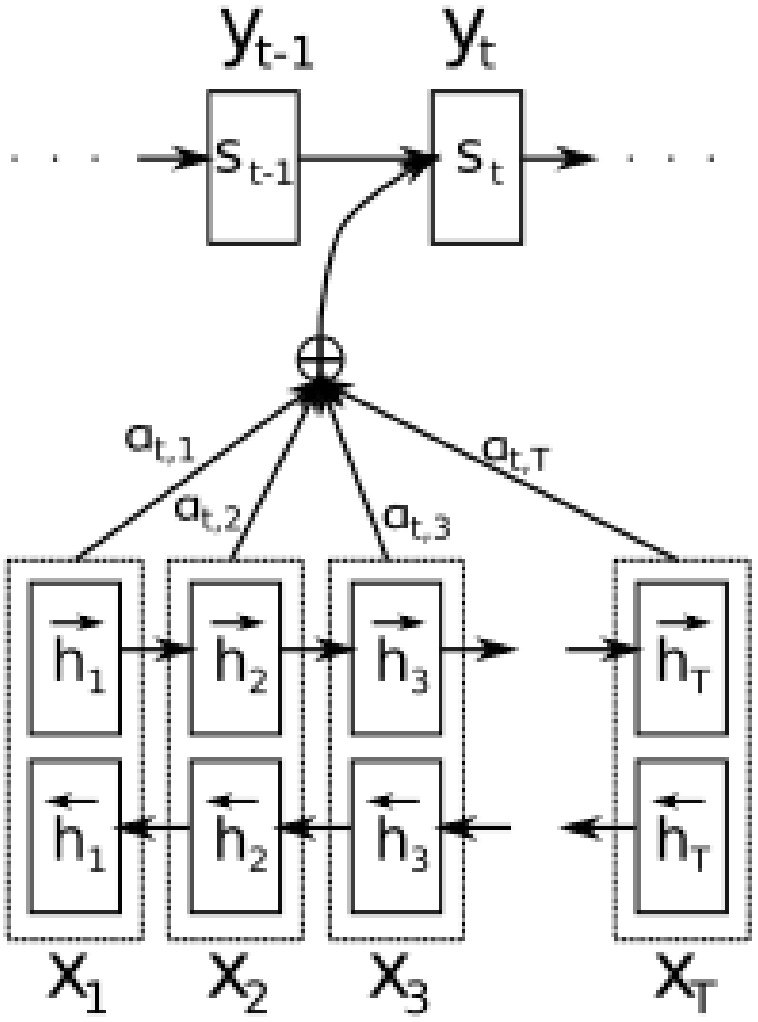


Gambar II.2: Ilustrasi dari arsitektur *encoder-decoder* (Cho dkk., 2014a)

encoder-decoder ini harus dapat memampatkan informasi dari satu kalimat utuh menjadi sebuah *fixed-length vector*. Hal ini bisa jadi menyulitkan RNN untuk merepresentasikan kalimat yang panjang, terutama jika panjangnya lebih panjang dari yang terdapat pada data latih di corpus. Cho dkk. (2014b) menunjukkan bahwa memang performa dari model *encoder-decoder*-nya semakin memburuk seiring dengan semakin meningkatnya panjang dari kalimat masukan.

Untuk menangani masalah tersebut, Bahdanau dkk. (2015) mengaplikasikan mekanisme *attention* pada *decoder*. Dengan mekanisme *attention* ini *decoder* dapat menentukan bagian mana dari kalimat masukan yang harus mendapatkan perhatian lebih untuk selanjutnya menjadi masukan pada fungsi aktivasi pada *decoder* untuk menentukan keluaran apa yang akan dihasilkan. Dengan membiarkan *decoder* menentukan masukan mana yang penting untuk menghasilkan keluaran selanjutnya, *decoder* menjadi tidak perlu untuk merepresentasikan semua informasi yang ada

dalam sebuah kalimat ke dalam sebuah vektor. Dengan ini informasi pada sebuah kalimat bisa direpresentasikan secara tersebar ke seluruh rangkaian kata-kata pada kalimat, yang selanjutnya bisa dipilih oleh *decoder* dengan mekanisme *attention*.



Gambar II.3: Ilurstrasi mekanisme attention (Bahdanau dkk., 2015)

Seperti yang bisa dilihat pada gambar II.3, tingkat kepentingan dari masukan dipilih oleh mekanisme *attention* berdasarkan nilai dari bobot $\alpha_{i,j}$ dari setiap masukan h_j, yang dihitung dengan fungsi *softmax* (Bahdanau dkk., 2015):

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{Tx} exp(e_{ik})}$$

yang dalam hal ini,

$$e_{ij} = a(s_{i-1}, h_j)$$

yang merupakan model *alignment* yang menilai kecocokan antara masukan pada sekitaran posisi j dan keluaran pada posisi i. Parameter pada model *alignment* dilatih sebagai *feedforward neural network* yang kemudian dilatih secara bersama-sama dengan komponen-komponen lain pada sistem yang diusulkan.

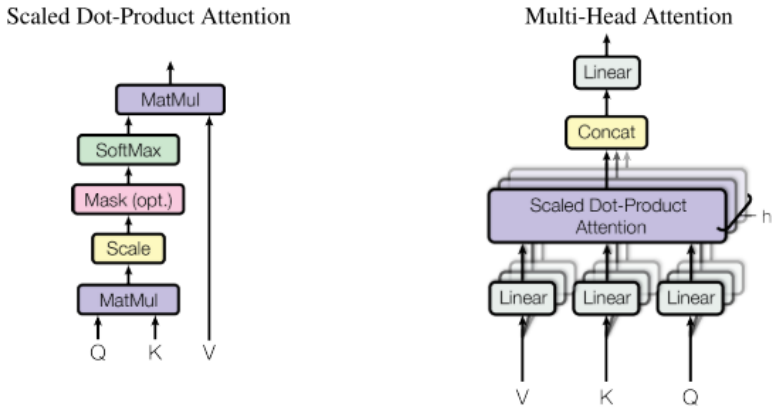
II.2.2 Model *Encoder-Decoder* berbasis *attention*

Mekanisme *attention* sudah diaplikasikan pada berbagai permasalahan selain permasalahan mesin translasi dan menjadi bagian penting dalam pemodelan rangkaian dan model transduksi, sehingga memungkinkan model untuk memodelkan dependensi tanpa harus melihat jarak antara masukan dan keluaran. Akan tetapi, kebanyakan pengaplikasian mekanisme *attention* ini hanya sebatas digunakan sebagai pelengkap untuk jaringan saraf rekuren. Oleh sebab itu Vaswani dkk. (2017) mengusulkan model yang disebut sebagai model transformer, sebuah arsitektur model yang menghindari penggunaan rekurens dan bergantung sepenuhnya pada mekanisme *attention* untuk menggambarkan dependensi global antara masukan dan keluaran. Selain itu model transformer ini memungkinkan dilakukannya paralelisasi sehingga dapat mempercepat proses pelatihan model, dan juga berhasil mengungguli model *encoder-decoder* berbasis rekurens.

Arsitektur dari model transformer secara keseluruhan mengikuti model *encoder-decoder*, hanya saja komponen penyusunnya tidak menggunakan RNN tapi menggunakan *stacked self-attention*, dan *point-wise fully connected layer*. Selain itu untuk model transfromer juga tidak menggunakan RNN untuk meng-*encode* rangkaian, tetapi menggunakan layer *positional encodings* yang kemudian diikuti oleh fungsi *attention*.

Fungsi *attention* bisa dideskripsikan sebagai pemetaan *query* dan sekumpulan pasangan *key-value* menjadi sebuah keluaran, yang pada hal ini *query*, pasangan *key-*

value, dan keluaran semuanya berbentuk vektor. Keluaran tersebut dihitung sebagai jumlah tertimbang. Jenis fungsi *attention* yang digunakan pada riset ini disebut sebagai *Scaled Dot-Product Attention*, yang kemudian fungsi *attention* tersebut bisa dihitung secara paralel, yang kemudian disebut sebagai *Multi-Head Attention*.



Gambar II.4: Ilustrasi *scaled dot-product attention* dan *multi-head attention*. (Vaswani dkk., 2017)

Scaled dot-product attention, mengambil input berupa beberapa *query* (Q) dan *key* (K) yang berdimensi d_k, dan *value* (V) yang berdimensi d_v, dan dapat diformulasikan sebagai berikut:

$$Attention(Q,K,V) = \frac{QK^T}{\sqrt{d_k}}V$$

Pada dasarnya *scaled dot-product attention* sama seperti *dot-product attention*, perbedaannya terletak pada penambahan faktor penskala $\frac{1}{d_k}$ pada perhitungannya. faktor penskala tersebut digunakan untuk menangani masalah ketika nilai d_k terlalu besar sehingga hasil *dot-product*nya tumbuh secara besar, sehingga hasil dari fungsi softmaxnya memiliki gradien yang sangat kecil dan selanjutnya menimbulkan masalah ketika melakukan propagasi balik.

Multi-head attention merupakan pengembangan dari *scaled dot-product attention*. Menurut Vaswani dkk. (2017), daripada menghitung satu kali *attention* dari *query*, *key*, dan *value* yang sebanyak d_{model}, akan lebih baik jika *query*, *key*, dan *value*nya

diproyeksikan secara linier terlebih dahulu sebanyak h kali, dengan proyeksi liner ke dimensi d_q, d_k, d_v dan yang berbeda-beda dan dipelajari dari data. Masing-masing *query*, *key*, dan *value* yang sudah diproyeksikan tersebut kemudian dihitung *attention*nya secara paralel, dan menghasilkan output berdimensi d_v yang kemudian dikonkatenasi dan diproyeksikan lagi, yang merupakan hasil akhir dari perhitungan *multi-head attention*. *Multi-head attention* dapat diformulasikan sebagai berikut:

$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

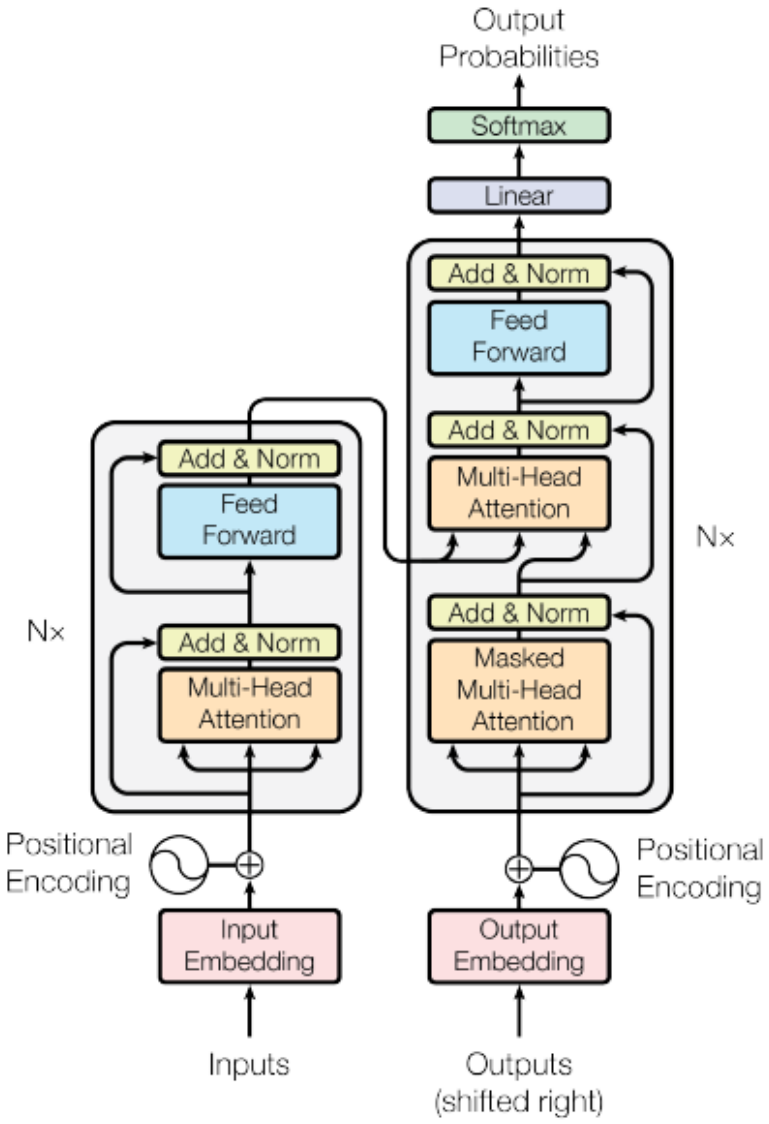
yang dalam hal ini QW_i^Q, KW_i^K, dan VW_i^V merupakan matrix parameter untuk proyeksi, yang dipelajari dari data.

Jenis *attention* ini memungkinkan model untuk mempelajari informasi mana yang paling perlu diperhatikan dari berbagai representasi pada ruang pencarian dengan posisi yang berbeda-beda.

Selain menggunakan fungsi *attention*, setiap layer pada *encoder* dan *decoder* berisi *fully connected feedforward network*, yang terdiri dari dua transformasi linier dengan fungsi aktivasi ReLU diantaranya. Di layer masukan terdapat layer *embedding* dan di layer keluaran digunakan layer softmax. Karena modelnya tidak mempunyai komponen rekurens atau konvolusi, supaya modelnya mampu mempelajari urutan dari rangkaiannya, maka modelnya harus dimasukkan informasi tambahan mengenai posisi relatif dan posisi absolut dari token di dalam sebuah kalimat. Untuk itu digunakan *positional embedding* setelah melalui *embedding* di layer masukan tadi.

II.3 Pemodelan Gambar dan Video

Riset yang paling banyak mengenai pemodelan gambar dan video adalah mengenai pembangkitan keterangan otomatis (*automatic caption generation*). Pembangkitan keterangan otomatis merupakan permasalahan mendasar pada kecerdasan buatan yang menggabungkan *computer vision* dan pemrosesan bahasa alami. Riset mengenai pembangkitan keterangan otomatis akhir-akhir ini bisa dikategorisasikan



Gambar II.5: Arsitektur model Transformer (Vaswani dkk., 2017)

menjadi dua, yaitu pembangkitan keterangan otomatis dari citra dan pembangkitan keterangan otomatis dari video. Jika proses pembangkitan dilakukan dari citra maka informasi penting yang harus diperhatikan hanya informasi posisi saja sedangkan jika dilakukan dari video maka selain itu harus juga diperhatikan informasi temporal antar frame videonya. Permasalahan utamanya adalah sebuah deskripsi dihasilkan harus bisa menangkap semua objek yang terdapat di dalam citra tersebut, dan juga harus bisa mengekspresikan keterkaitan antara objek dan juga atribut-atribut apa saja yang menjelaskan objek tersebut dan aktivitas apa yang melibatkan objek tersebut. Terlebih lagi, deskripsi tersebut harus dideskripsikan dengan menggunakan

bahasa yang sealami mungkin.

II.3.1 Pembangkitan Keterangan Otomatis dari Citra

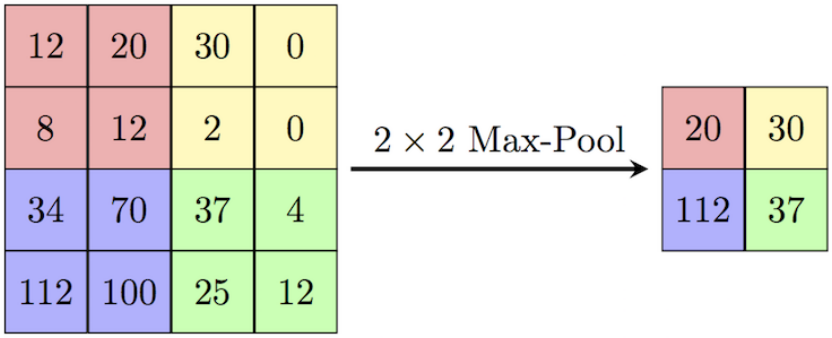
Riset Vinyals dkk. (2014) mengemukakan sebuah model yang diinspirasi oleh kemajuan-kemajuan terbaru pada permasalahan mesin translasi yang menggunakan model *encoder-decoder* berbasis RNN. Perbedaannya pada riset ini adalah sebagai ganti dari RNN, modelnya menggunakan *convolutional neural network* (CNN) sebagai encoder. Dalam beberapa tahun terakhir penggunaan CNN telah menunjukkan hasil yang baik dalam merepresentasikan sebuah citra menjadi sebuah *fixed-length vector*, sehingga selanjutnya bisa digunakan sebagai masukan dari *decoder* RNN untuk menghasilkan keluaran berupa kalimat deskripsi. /bigskip

CNN merupakan bagian dari *deep feedforward artificial neural networks* yang pada umumnya digunakan untuk menganalisis citra. CNN merupakan variasi dari *multilayer perceptron* dan didesain supaya memerlukan praproses seminimal mungkin. CNN terdiri dari banyak lapisan tersembunyi yang melakukan konvolusi dan *pooling* terhadap masukan yang pada umumnya berbentuk citra. Lapisan-lapisan konvolusi dan *pooling* ini memiliki nilai-nilai parameter yang dipelajari dari data sehingga Lapisan-lapisan tersebut secara otomatis menyesuaikan untuk bisa mengekstraksi informasi yang paling penting. Pada CNN, lapisan tersembunyinya biasanya terdiri atas lapisan konvolusi, lapisan *pooling*, *fully connected layers* dan lapisan normalisasi.

Lapisan konvolusi menggunakan operasi konvolusi kepada masukan, dan meneruskan hasilnya ke lapisan selanjutnya. Setiap lapisan konvolusi memproses data hanya pada *receptive field*nya. *Receptive field* adalah sebagian area dari keseluruhan data yang akan diproses. Area tersebut biasanya berbentuk persegi. *Receptive field* digunakan untuk mengurangi jumlah parameter jika dibandingkan dengan menggunakan *fully connected layer*, karena ukuran dari sebuah citra yang biasanya berukuran besar, dan setiap pixelnya merupakan input yang relevan.

Lapisan *pooling* pada CNN bertugas untuk menggabungkan kluster neuron dari lapisan keluaran sebelumnya menjadi satu neuron pada layer selanjutnya. Misalnya,

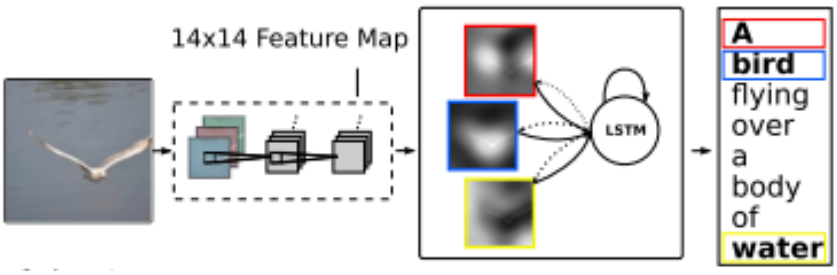
pada lapisan m*max-pooling* diambil nilai maksimum dari setiap kluster di lapisan sebelumnya. Contoh lainnya adalah lapisan *average-pooling* yang mengambil nilai rata-rata dari setiap kluster di lapisan sebelumnya.



Gambar II.6: Ilustrasi lapisan *max-pooling*

Fully connected layer cara kerjanya sama seperti *multilayer perceptron*, menghubungkan setiap neuron pada lapisan sebelumnya ke setiap neuron yang ada pada lapisan setelah lapisan tersebut.

Diinspirasi lebih lanjut dari permasalahan mesin translasi yang menggunakan mekanisme *attention*, riset Xu dkk. (2015) mengembangkan lebih lanjut riset Vinyals dkk. (2014) dengan memberikan mekanisme *attention* pada saat *decoder* melakukan pembangkitan deskripsi dari citra.

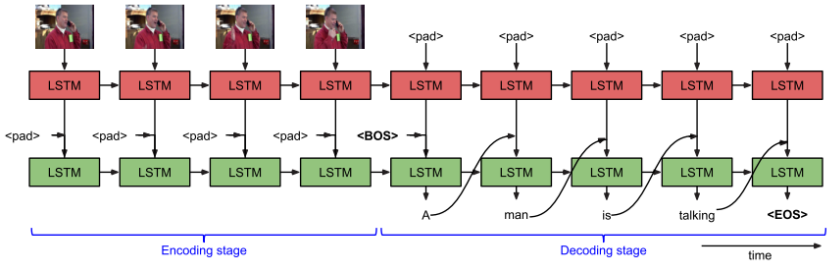


Gambar II.7: Ilustrasi model *encoder-decoder* dengan menggunakan *attention* pada *image captioning* (Xu dkk., 2015).

II.3.2 Pembangkitan Keterangan Otomatis dari Video

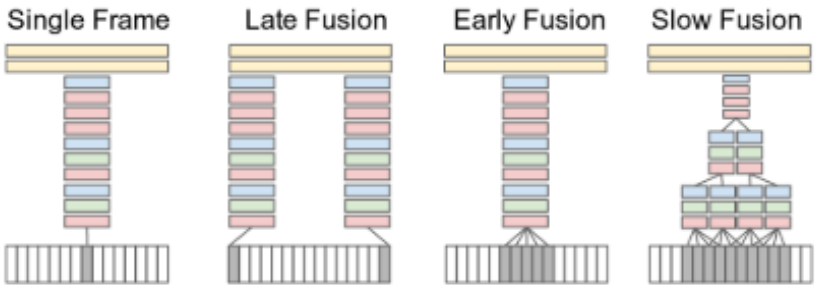
Riset lain yang juga terinspirasi dari perkembangan terkini dalam mesin translasi adalah riset mengenai pembangkitan keterangan otomatis dari video yang dilakukan oleh Venugopalan dkk. (2015). Model yang dikemukakan menggunakan model

seqence-to-sequence dengan *recurrent network* yang digunakan sebagai *decoder* dan *encoder*-nya adalah LSTM bertumpuk. Masukan dari LSTM bertumpuk tersebut berupa citra RGB dan informasi *optical flow* yang kemudian keduanya direpresentasikan menjadi sebuah *embedding* dengan menggunakan CNN yang parameternya dilatih bersama-sama dengan parameter yang terdapat pada LSTM bertumpuk.



Gambar II.8: Ilustrasi model *video-to-text* (Chung dkk., 2017).

Selain itu ada riset lain yang tidak menggunakan *recurrent network* tetapi menggunakan CNN yang telah dimodifikasi untuk dapat memanfaatkan informasi spasial dan temporal pada video, sehingga diberi nama *spatio-temporal convolutional neural network* (STCNN) (Karpathy dkk., 2014). Cara kerjanya sama seperti CNN hanya saja keterhubungan jaringannya diperluas pada dimensi waktunya sehingga model dapat mempelajari fitur spatio-temporal. Perluasannya tersebut dibagi menjadi tiga, yaitu *early fusion*, *late fusion*, dan *slow fusion*.



Gambar II.9: Ilustrasi STCNN (Karpathy dkk., 2014).

Pada *early fusion*, modelnya menggabungkan informasi temporal di seluruh frame dengan *window* yang telah ditentukan, langsung pada tingkat pixel. Sedangkan pada *late fusion*, dari *window* yang telah ditentukan diambil dua buah frame, frame

awal dan frame akhir, lalu masing-masing frame melalui berbagai lapisan konvolusi, lalu menggabungkan kedua representasi yang didapatkan dengan menggunakan *fully connected layer*. Untuk *slow fusion*, merupakan model yang menggabungkan dua pendekatan tadi (*early fusion* dan *late fusion*) dengan cara menggabungkan informasi temporal dari semua frame pada *window* secara perlahan-lahan dan hirarkis.

II.4 *Visual Speech Recognition*

Riset Chung dkk. (2017) melakukan task pembangkitan keterangan otomatis dari video yang cakupannya lebih kecil, yaitu melakukan pengenalan gerak bibir untuk *visual speech recognition*. Model yang digunakan dapat dibagi menjadi tiga modul, yaitu modul *Watch*, modul *Listen*, modul *Attend*, dan modul *Spell*, dan bisa diformalisasikan sebagai berikut (Chung dkk., 2017):

$$s^v, o^v = Watch(x^v)$$
$$s^a, o^a = Listen(x^a)$$
$$P(y|x^v, x^a) = Spell(s^v, s^a, o^v, o^a)$$

Modul *Watch*, merupakan *encoder* citra yang terdiri atas modul konvolusi yang menghasilkan fitur citra f_i^v untuk setiap masukan di *time-step* x_i^v, ditambah dengan sebuah modul rekuren yang menghasilkan fixed-length vector s^v dan sekumpulan vektor output o^v.

Modul *Listen*, merupakan *encoder* yang berisi sama seperti modul *Watch* tetapi tanpa memiliki modul konvolusi. Modul rekurennya menerima masukan berupa hasil ekstraksi fitur menggunakan MFCC berdimensi 13, lalu dari modul rekuren tersebut dihasilkan vektor berukuran tetap s^a dan sekumpulan vektor keluaran o^a.

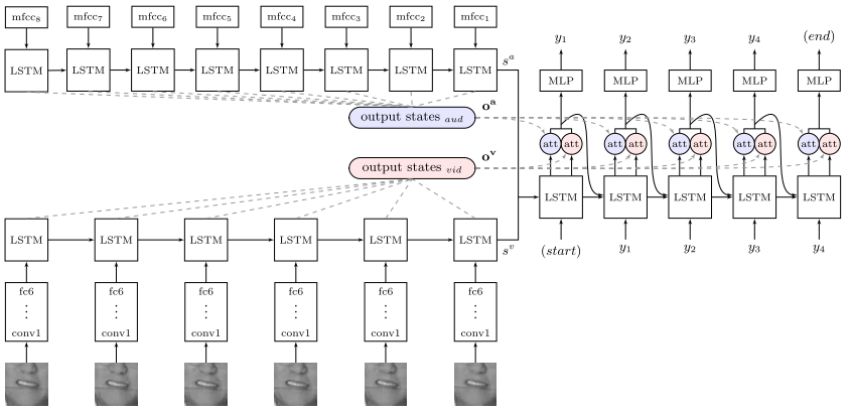
Modul *Spell*, berbasis pada transduser LSTM dan menghasilkan keluaran berupa rangkaian token pada level karakter. Modul ini bisa diformulasikan sebagai berikut (Chung dkk., 2017).

$$h_k^d, h_k^d = LSTM(h_{k-1}^d, y_{k-1}, c_{k-1}^v, c_{k-1}^a)$$

$$c_k^v = o^v \cdot Attention^v(h_k^d, o^v)$$

$$c_k^a = o^a \cdot Attention^a(h_k^d, o^a)$$

$$P(y_i|x^v, x^a, y_{<i}) = softmax(MLP(o_k^d, c_k^v, c_k^a))$$



Gambar II.10: Arsitektur model *Watch*, *Listen*, *Attend*, dan *Spell* (Chung dkk., 2017).