COMP7220/8220: A Sampling of More Advanced Topics

Mark Dras

June 2020

Outline

Encoder-Decoder for Machine Translation

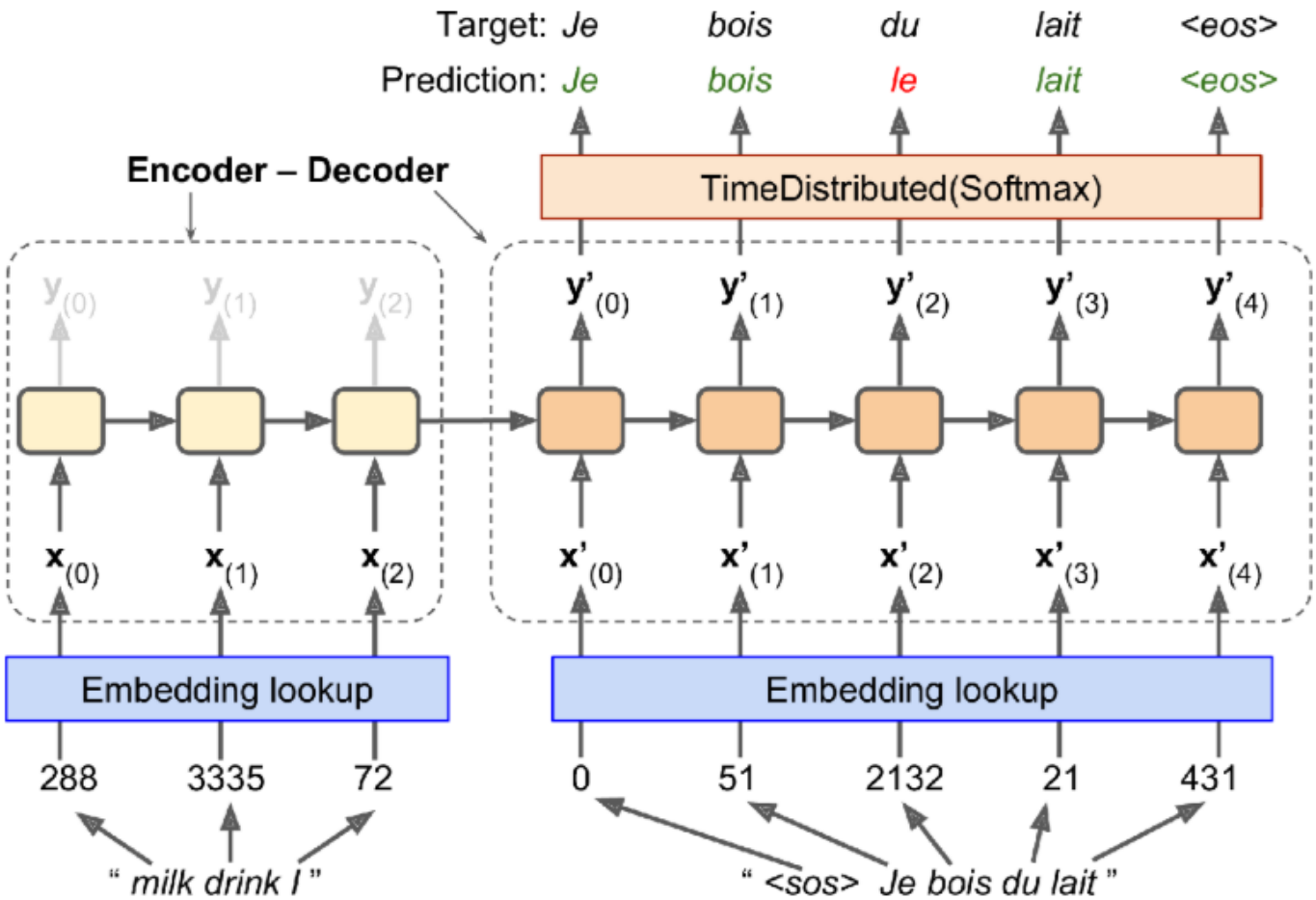


Figure 16-3. A simple machine translation model

Machine Translation Models

Considerations

- Varying sentence lengths.
  - Can vary a lot, so could need lots of padding.
  - Can't just throw parts away (as in e.g. sentiment analysis).

  $\Rightarrow$ Put into length buckets and pad within buckets.
- When the output vocabulary is large (which is the case here), outputting a probability for each and every possible word would be terribly slow.

  $\Rightarrow$ Use SAMPLED SOFTMAX: look only at output logits for correct word and random sample of incorrect words.

Bidirectional RNNs

In MT

Want both left and right context of words (e.g. “free drinks” vs “free people”).

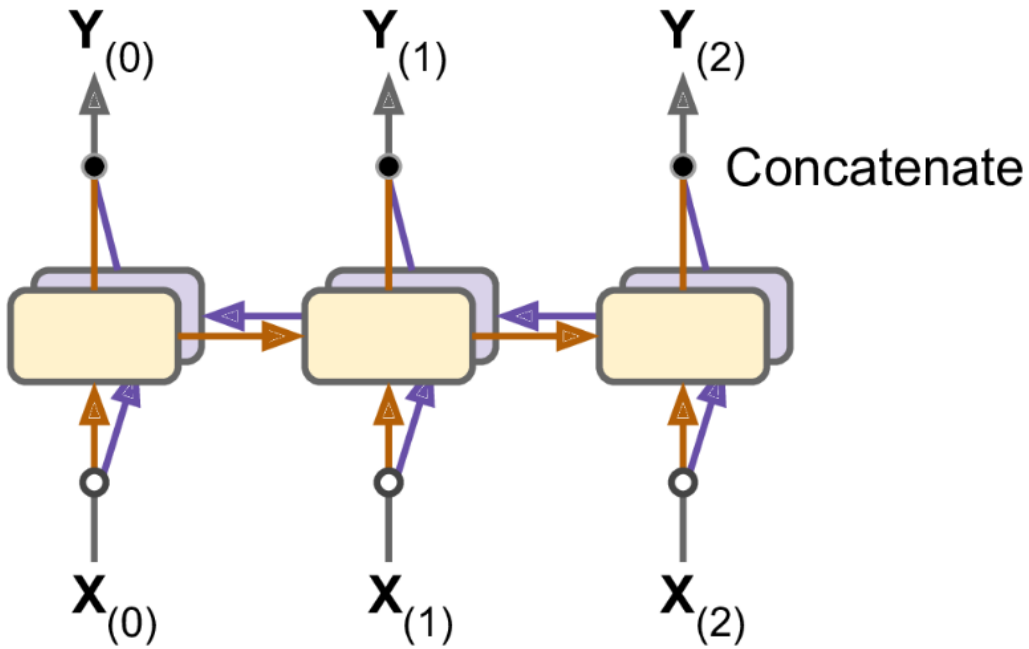


Figure 16-5. A bidirectional recurrent layer

Keras

```
keras.layers.Bidirectional(keras.layers.GRU(10, return_sequences=True))
```

Attention Mechanisms (1)

Idea

- Long distance between corresponding source and target words.
  - "milk", "lait" in example.
  - Eng-Ger: "As Gregor Samsa **awoke** one morning from uneasy dreams he found himself **transformed** in his bed into a gigantic insect." $\longleftrightarrow$ "Als Gregor Samsa eines Morgens aus unruhigen Träumen **erwachte**, fand er sich in seinem Bett zu einem ungeheueren Ungeziefer **verwandelt**."
- Use ATTENTION to focus on relevant source words when decoding.
  - Encoder sends all outputs to decoder, not just final state.
  - Learn weights over these.

Attention Mechanisms (2)

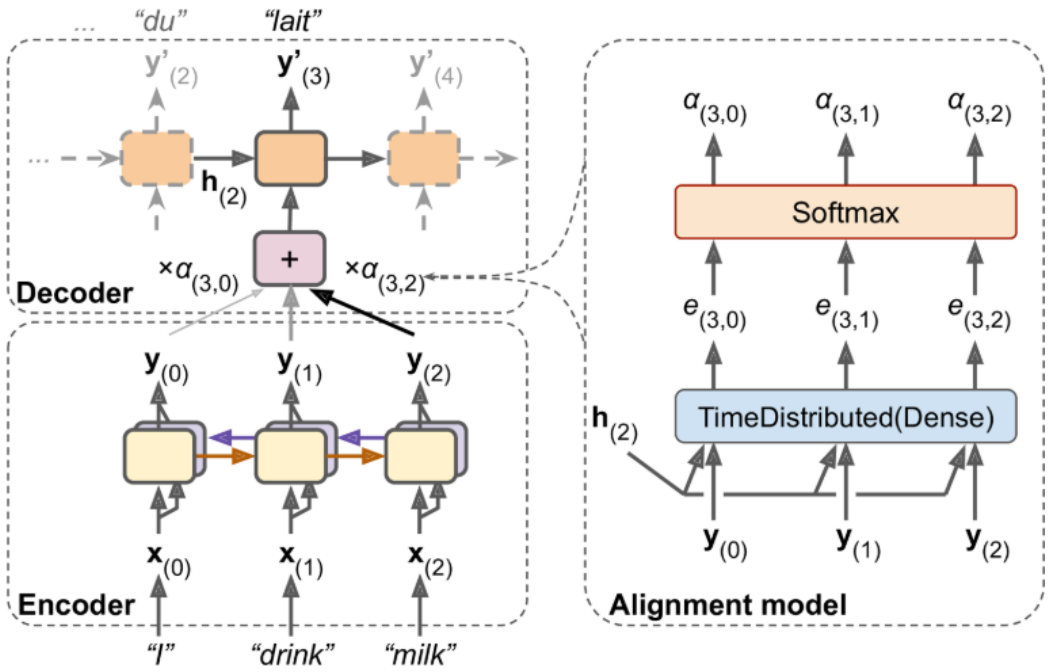


Figure 16-6. Neural machine translation using an Encoder–Decoder network with an attention model

Attention Weights

$$\tilde{\mathbf{h}}_{(t)} = \sum_i \alpha_{(t,i)} \mathbf{y}_i$$

$$\alpha_{(t,i)} = \frac{\exp(e_{(t,i)})}{\sum_{i'} \exp(e_{(t,i')})}$$

$$e_{(t,i)} = \mathbf{h}_{(t)}^T \mathbf{y}_i$$

Visual Attention

Figure 16-7. Visual attention: an input image (left) and the model's focus before producing the word "frisbee" (right)[18]

The Transformer (1)

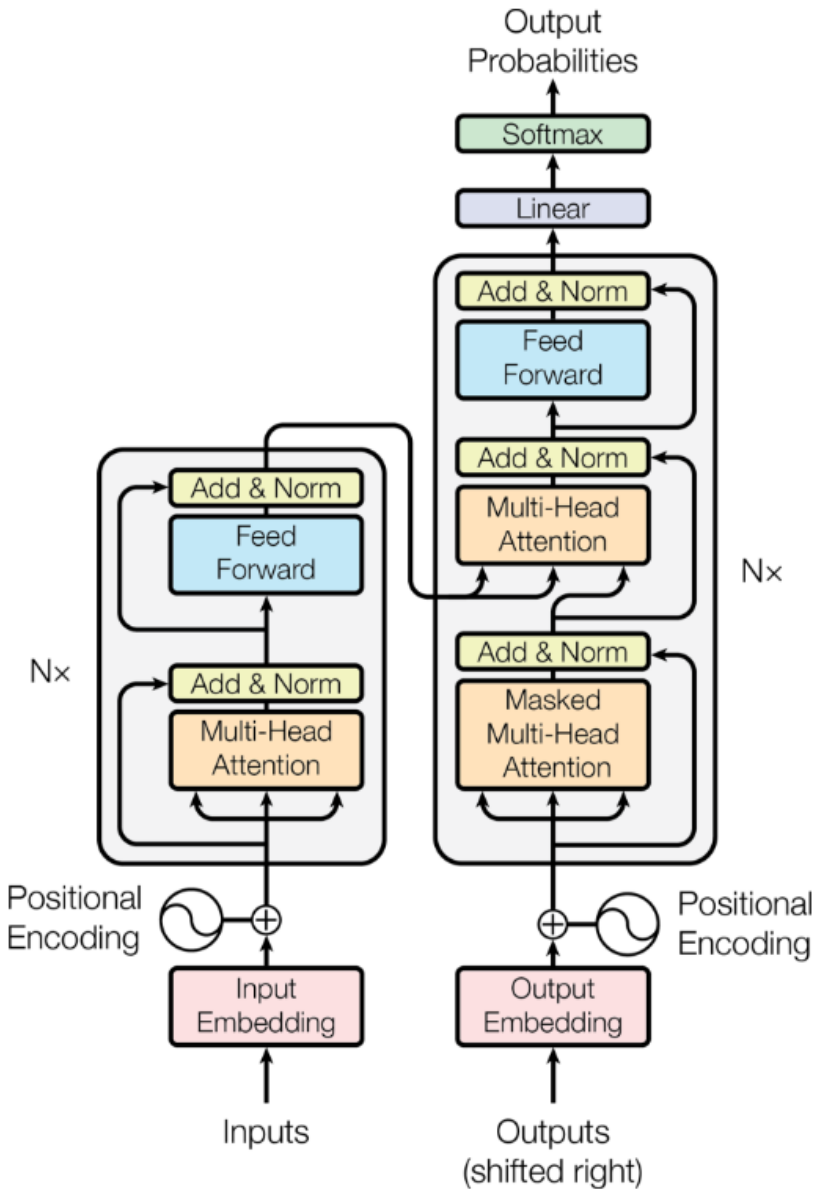


Figure 16-8. The Transformer architecture[22]

The Transformer (2)

Noteworthy Characteristics

- Positional encoding.
- Scaled Dot-Product Attention.
  - Consider "They played chess"; have generated "they", want to generate a verb.
  - Like a fuzzy dictionary lookup: the QUERY.

$$\mathit{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{keys}}}\right)\mathbf{V}$$

- Multihead Attention: multiple of these.

The Transformer (3)

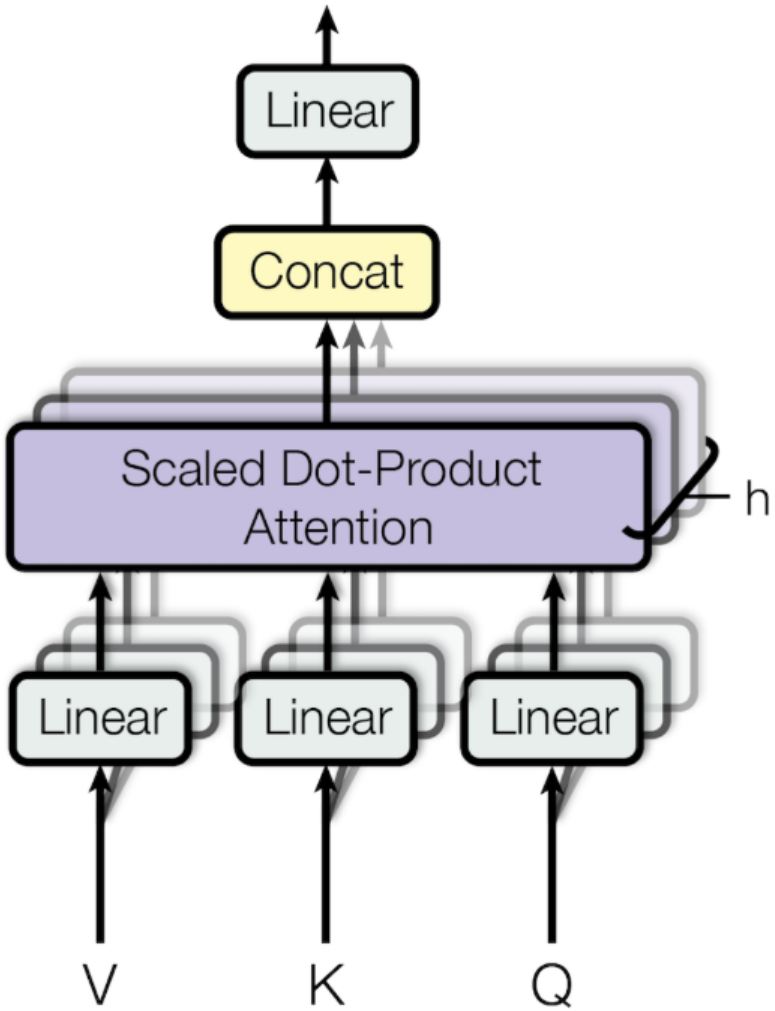


Figure 16-10. Multi-Head Attention layer architecture[23]

Outline

Language Model Innovations (1)

ELMo

Embeddings from Language Models (ELMo): these are contextualized word embeddings learned from the internal states of a deep bidirectional language model. For example, the word "queen" will not have the same embedding in "Queen of the United Kingdom" and in "queen bee."

ULMFiT

Howard and Ruder trained an LSTM language model using self-supervised learning (i.e., generating the labels automatically from the data) on a huge text corpus, then they fine-tuned it on various tasks. Their model outperformed the state of the art on six text classification tasks by a large margin (reducing the error rate by 18–24% in most cases). Moreover, they showed that by fine-tuning the pretrained model on just 100 labeled examples, they could achieve the same performance as a model trained from scratch on 10,000 examples.

Language Model Innovations (2)

GPT2

Radford and other OpenAI researchers also demonstrated the effectiveness of unsupervised pretraining, but this time using a Transformer-like architecture. The authors pretrained a large but fairly simple architecture composed of a stack of 12 Transformer modules (using only Masked Multi-Head Attention layers) on a large dataset, once again trained using self-supervised learning. Then they fine-tuned it on various language tasks, using only minor adaptations for each task.

Evaluations

- GLUE Benchmark: `https://gluebenchmark.com/`
- Now-standard selection of NLP tasks.

GLUE

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

Table 1: Task descriptions and statistics. All tasks are single sentence or sentence pair classification, except STS-B, which is a regression task. MNLI has three classes; all other classification tasks have two. Test sets shown in bold use labels that have never been made public in any form.

Language Model Innovations (3)

BERT

Devlin and other Google researchers also demonstrates the effectiveness of self-supervised pretraining on a large corpus, using a similar architecture to GPT but non-masked Multi-Head Attention layers (like in the Transformer's encoder). This means that the model is naturally bidirectional; hence the B in BERT (Bidirectional Encoder Representations from Transformers). Two sources of strength:

- Masked language model (MLM).
- Next sentence prediction (NSP).

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard).

Outline

Australia's Winning Entry 2020

[Youtube: `https://youtu.be/sAzULywAHUM`]

[Discussion: `https://sites.rmit.edu.au/alu/2020/04/10/my-first-ai-song-contest-experience-its-not-a-bug-its-a-feature/`]

Some Mechanics

Team

Includes Uncanny Valley and RMIT.

Deep Learning

- MIDI melodies:
  - Modified Sample-RNN.
  - Trained on a 200 melody Eurovision dataset.
- Lyrics:
  - Modified GPT2.
  - Pretrained language model, fine-tuned on Eurovision lyrics.

Various Other Techniques

- Preprocessing melodies for MIDI generator.
- Converting lyric, melodies into stress patterns for pattern matching.
- Aligning melodies and lyrics.

Outline

Structure (1)

Idea

- Make neural networks compete against each other in the hope that this competition will push them to excel.
- Has two key components: generator and discriminator.

Generator

Takes a random distribution as input (typically Gaussian) and outputs some data—typically, an image. You can think of the random inputs as the latent representations (i.e., codings) of the image to be generated. So, as you can see, the generator offers the same functionality as a decoder in a variational autoencoder, and it can be used in the same way to generate new images (just feed it some Gaussian noise, and it outputs a brand-new image).

Discriminator

Takes either a fake image from the generator or a real image from the training set as input, and must guess whether the input image is fake or real.

Structure (2)

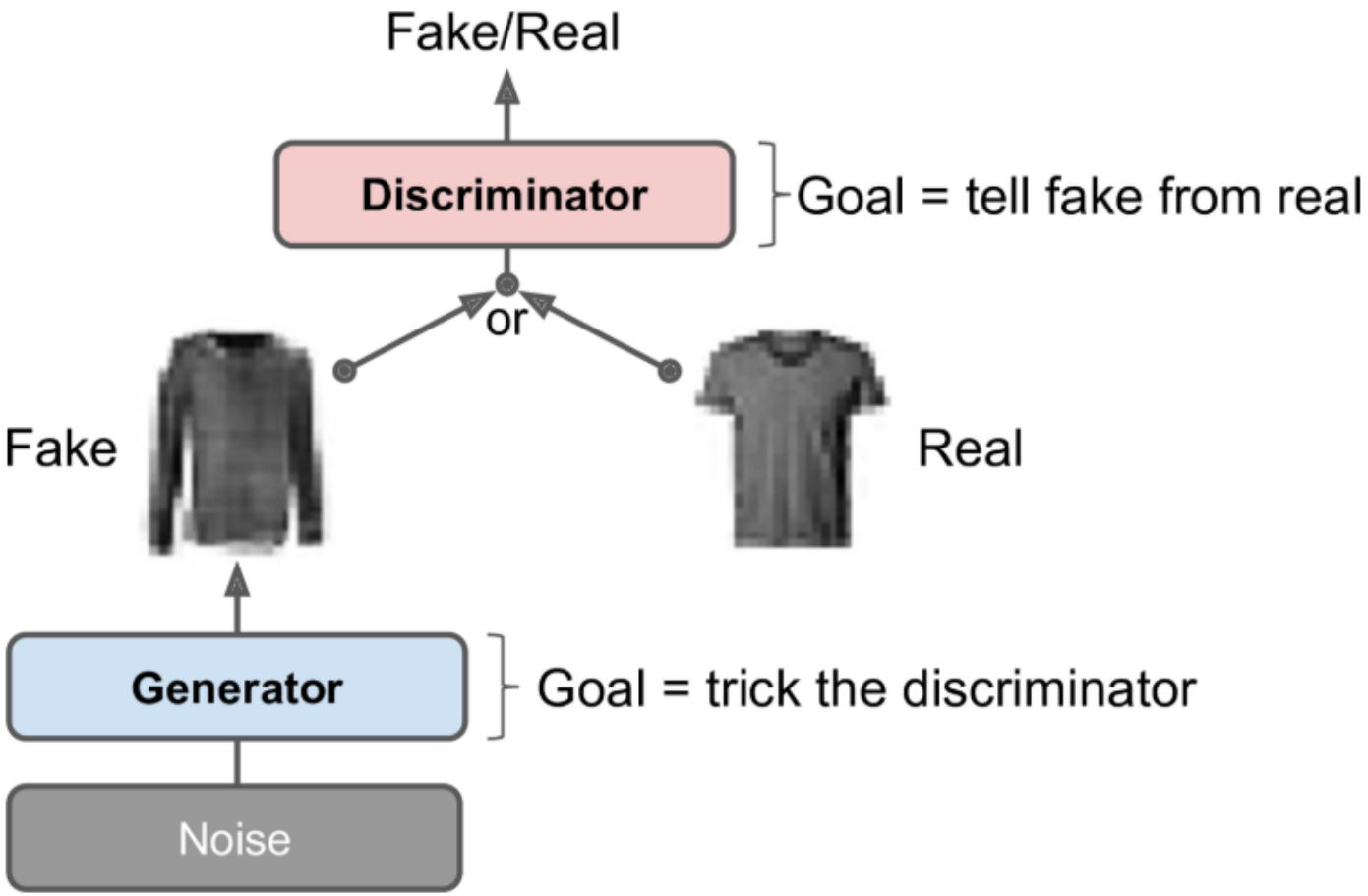


Figure 17-15. A generative adversarial network

Example Code for Fashion MNIST (1)

Model Definition

```
codings_size = 30

generator = keras.models.Sequential([
    keras.layers.Dense(100, activation="selu", input_shape=[codings_size]),
    keras.layers.Dense(150, activation="selu"),
    keras.layers.Dense(28 * 28, activation="sigmoid"),
    keras.layers.Reshape([28, 28])
])
discriminator = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(150, activation="selu"),
    keras.layers.Dense(100, activation="selu"),
    keras.layers.Dense(1, activation="sigmoid")
])
gan = keras.models.Sequential([generator, discriminator])

discriminator.compile(loss="binary_crossentropy", optimizer="rmsprop")
discriminator.trainable = False
gan.compile(loss="binary_crossentropy", optimizer="rmsprop")
```

Example Code for Fashion MNIST (2)

Training

```
batch_size = 32
dataset = tf.data.Dataset.from_tensor_slices(X_train).shuffle(1000)
dataset = dataset.batch(batch_size, drop_remainder=True).prefetch(1)

def train_gan(gan, dataset, batch_size, codings_size, n_epochs=50):
    generator, discriminator = gan.layers
    for epoch in range(n_epochs):
        for X_batch in dataset:
            # phase 1 - training the discriminator
            noise = tf.random.normal(shape=[batch_size, codings_size])
            generated_images = generator(noise)
            X_fake_and_real = tf.concat([generated_images, X_batch], axis=0)
            y1 = tf.constant([[0.]] * batch_size + [[1.]] * batch_size)
            discriminator.trainable = True
            discriminator.train_on_batch(X_fake_and_real, y1)
            # phase 2 - training the generator
            noise = tf.random.normal(shape=[batch_size, codings_size])
            y2 = tf.constant([[1.]] * batch_size)
            discriminator.trainable = False
            gan.train_on_batch(noise, y2)

train_gan(gan, dataset, batch_size, codings_size)
```

(Improved) GAN Output

Figure 17-17. Images generated by the DCGAN after 50 epochs of training

GAN Fakes

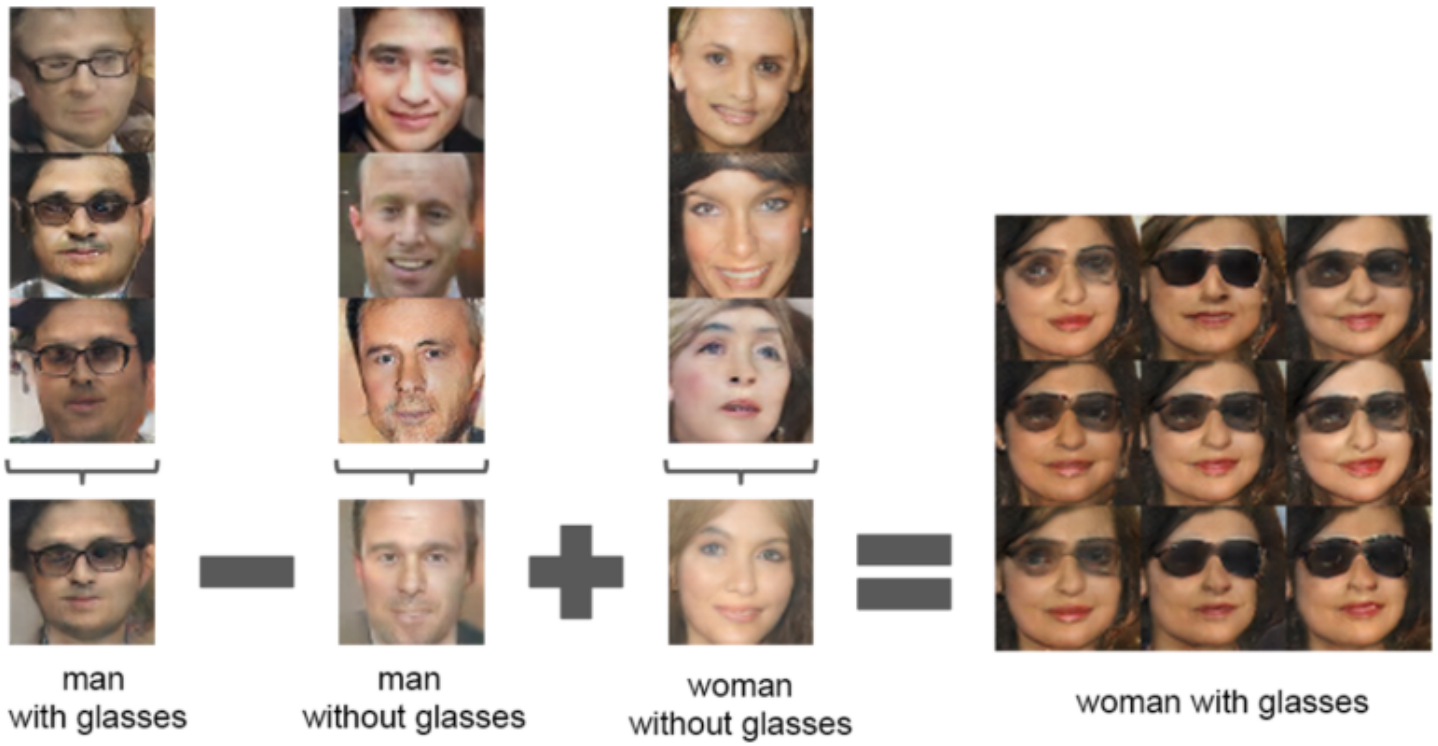


Figure 17-18. Vector arithmetic for visual concepts (part of figure 7 from the DCGAN paper)[14]

StyleGAN

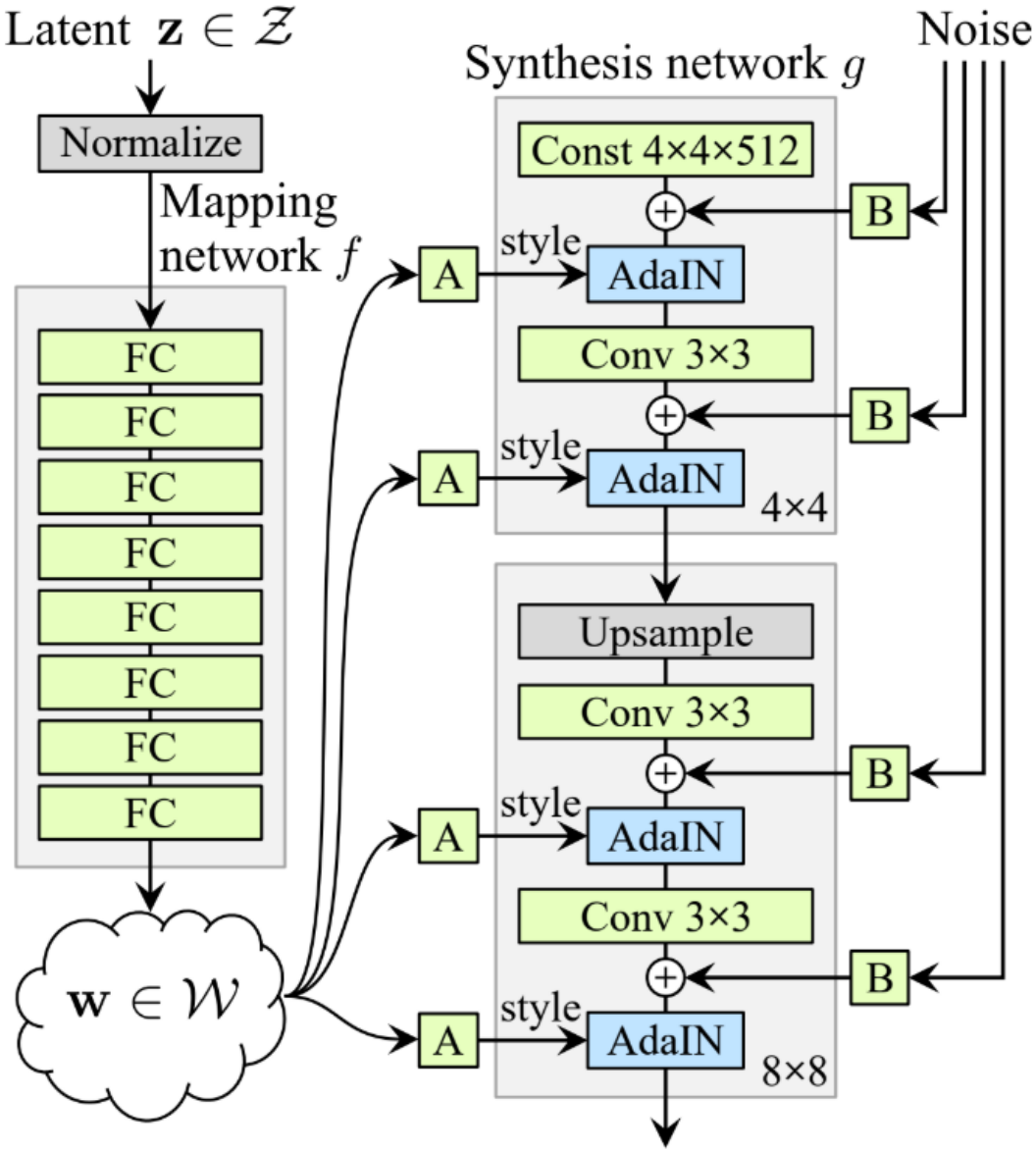
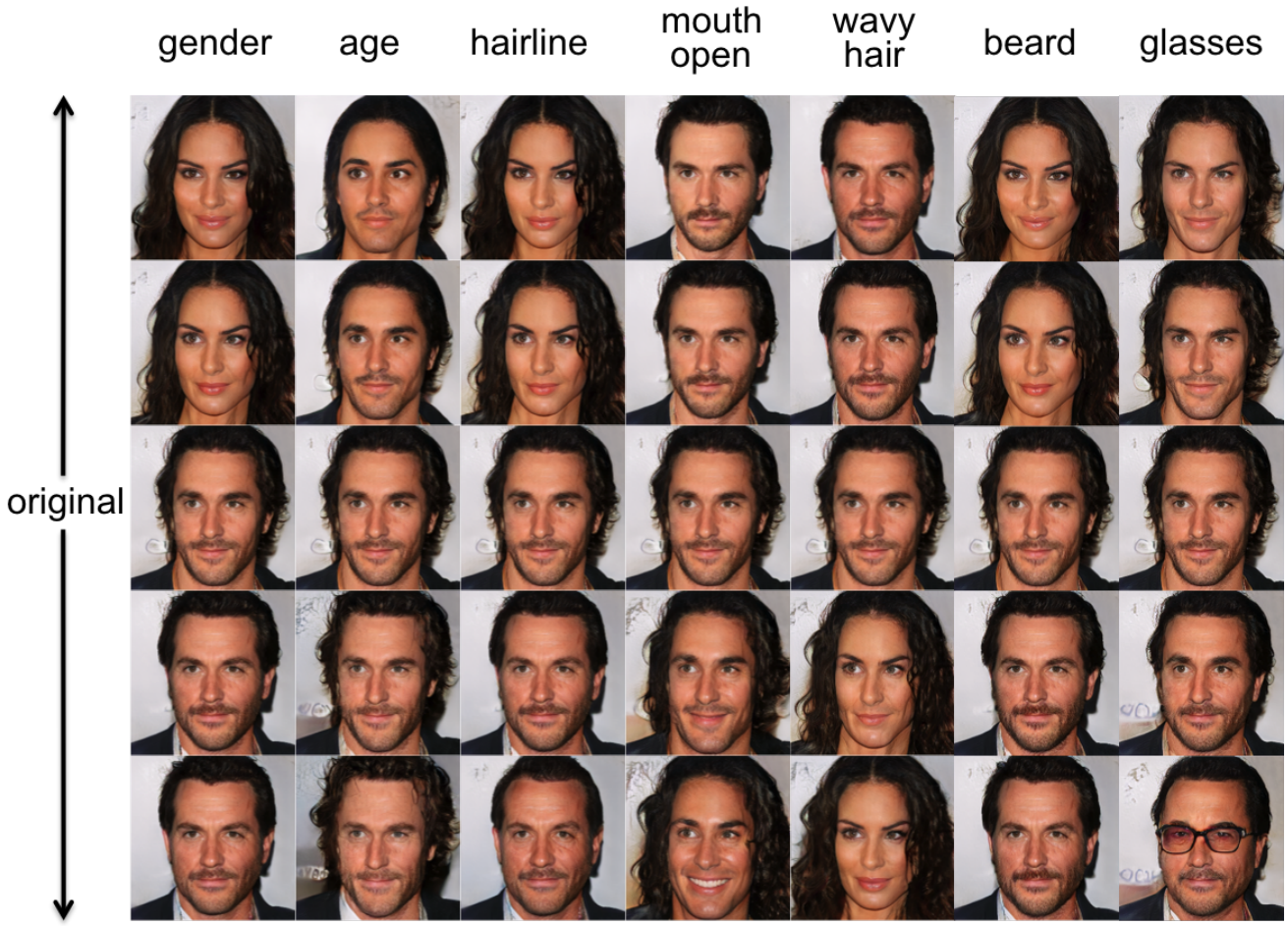


Figure 17-20. StyleGAN's generator architecture (part of figure 1 from the StyleGAN paper)[19]

Neural Style Transfer (1)



Neural Style Transfer (2)

illustration2vec (1)

Figure 2: *Qualitative results of the tag prediction. We show the top 5 tags that have the highest confidence scores.*

[paper: `https://www.gwern.net/docs/anime/2015-saito.pdf`]

[github: `https://github.com/rezoo/illustration2vec`]

illustration2vec (2)

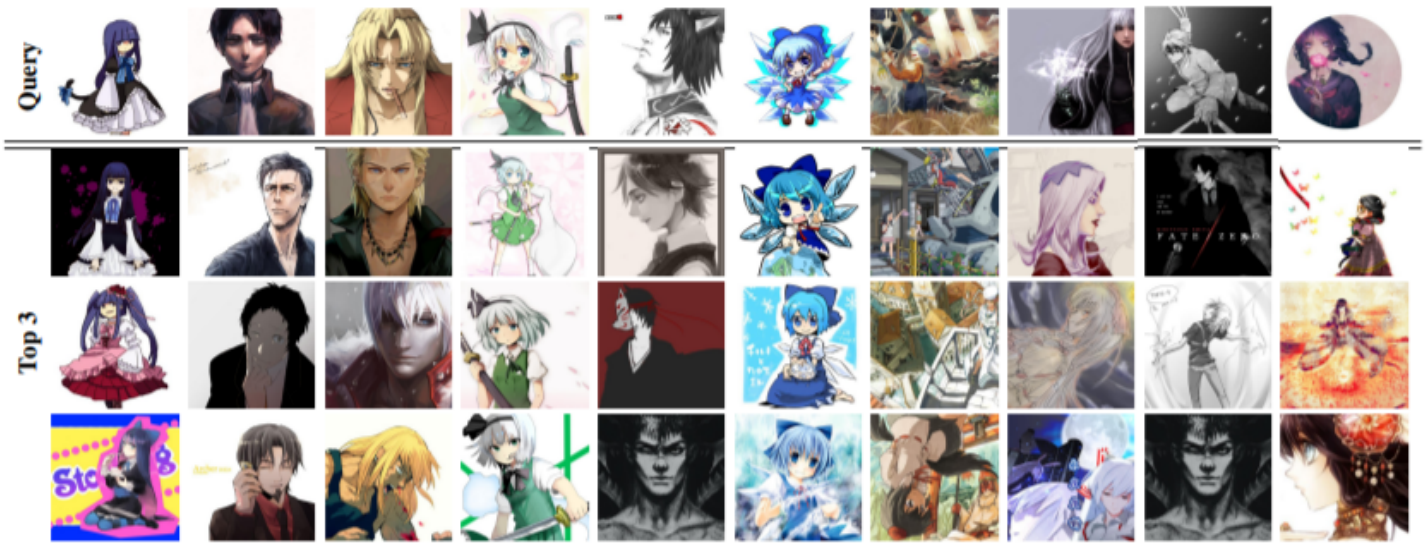


Figure 3: *Nearest neighbor retrieval: for each query, we show the top 3 illustrations from our test dataset. In order to indicate the source, we put all the illustration links into the above figure.*

TwinGAN

TwinGAN Fail

Failure case generated by Twin-GAN