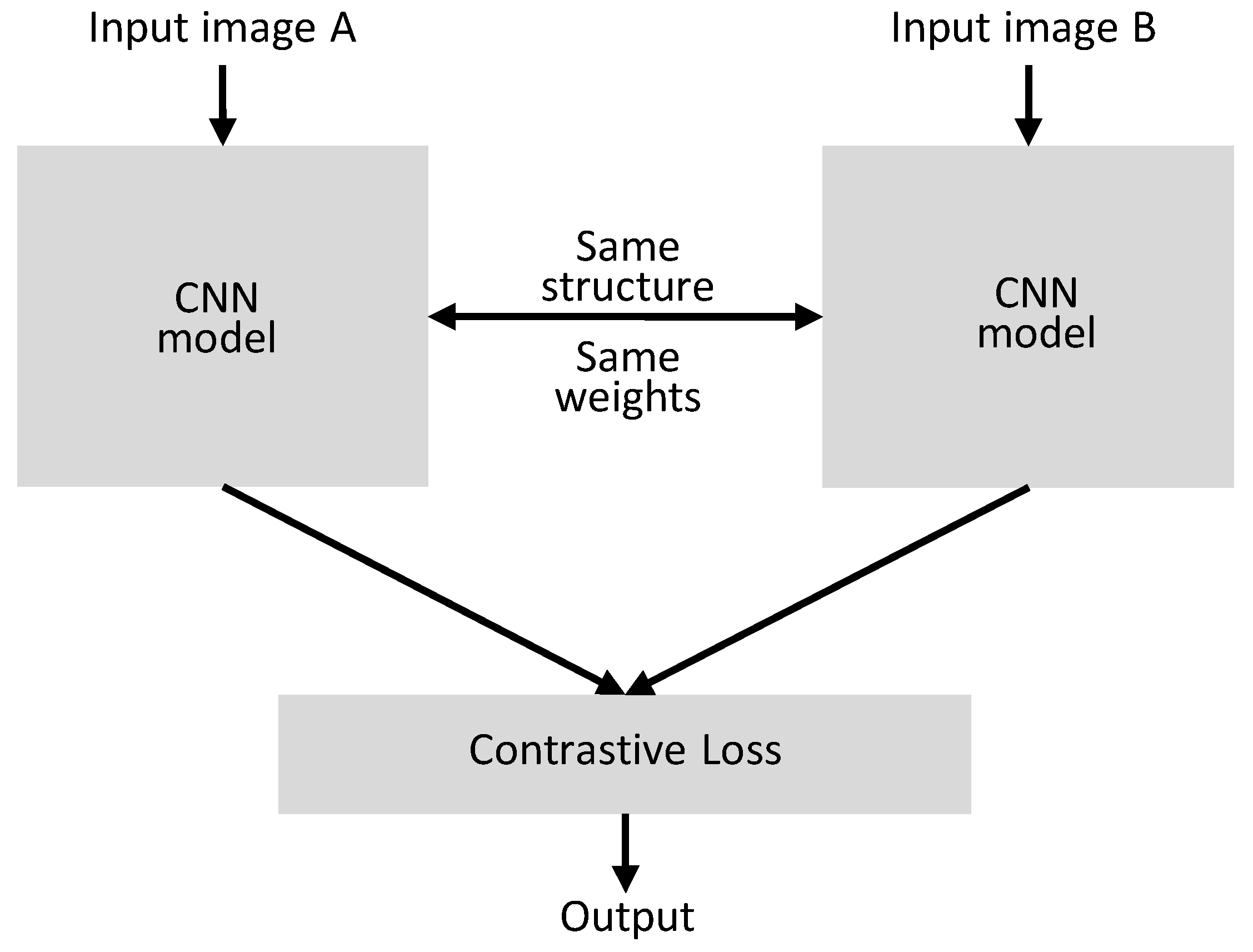
# Siamese Network

## Face Detection

### Model Description

The objective of Siamese network is to learn whether two input values are similar or dissimilar.



[Figure ] Siamese Model

But how? We can do this by using same feature extraction model and compare two features.

The model we build is face detection model. So CNN will be used as our feature extraction model. And we will use Euclidean distance as our energy function.

|  |  |  |
| --- | --- | --- |
| **Data type** | **Feature Extraction Model** | **Energy Function** |
| Image | CNN | Euclidean Distance |

[Figure ] Face Detection Model

### Data Preprocessing

Our datasets compose of 40 people. For each person, 10 picture are used as training data. Size of picture is resized into (56, 46).

|  |  |
| --- | --- |
| **Image size** | (56,46) |
| **The number of people** | 40 |
| **The number of picture for each person** | 10 |
| **Genuine Pairs** | 10,000 |
| **Fake Pairs** | 10,000 |

[Figure ] Dataset size

According to above table, our preprocessed dataset size is (20,000, 2, 56, 46, 1) (genuine, pair, height, width, gray-channel)

### Training Result

We train this model and here is result. We split train and test dataset as 0.75:0.25. This is train result. We add our face image into datasets. The test accuracy go down by about 0.1%. (0.97->0.96)

img\_1 = x\_train[:, 0]

img\_2 = x\_train[:, 1]

tensorboard = TensorBoard(log\_dir="log/{}".format(time()))

model.fit([img\_1, img\_2], y\_train, validation\_split=.25,

batch\_size=128, verbose=2, epochs=epochs, callbacks=[tensorboard])

Epoch 1/13

WARNING:tensorflow:From C:\Users\consi\anaconda3\lib\site-packages\tensorflow\python\ops\summary\_ops\_v2.py:1277: stop (from tensorflow.python.eager.profiler) is deprecated and will be removed after 2020-07-01.

Instructions for updating:

use `tf.profiler.experimental.stop` instead.

WARNING:tensorflow:Callbacks method `on\_train\_batch\_end` is slow compared to the batch time (batch time: 0.1629s vs `on\_train\_batch\_end` time: 0.3065s). Check your callbacks.

88/88 - 29s - loss: 0.1879 - val\_loss: 0.2543

Epoch 2/13

88/88 - 19s - loss: 0.1168 - val\_loss: 0.2060

Epoch 3/13

88/88 - 19s - loss: 0.0845 - val\_loss: 0.1003

Epoch 4/13

88/88 - 18s - loss: 0.0650 - val\_loss: 0.0713

Epoch 5/13

88/88 - 20s - loss: 0.0544 - val\_loss: 0.0513

Epoch 6/13

88/88 - 19s - loss: 0.0468 - val\_loss: 0.0459

Epoch 7/13

88/88 - 18s - loss: 0.0424 - val\_loss: 0.0431

Epoch 8/13

88/88 - 18s - loss: 0.0383 - val\_loss: 0.0329

Epoch 9/13

88/88 - 17s - loss: 0.0348 - val\_loss: 0.0275

Epoch 10/13

88/88 - 19s - loss: 0.0320 - val\_loss: 0.0224

Epoch 11/13

88/88 - 18s - loss: 0.0298 - val\_loss: 0.0183

Epoch 12/13

88/88 - 17s - loss: 0.0275 - val\_loss: 0.0210

Epoch 13/13

88/88 - 18s - loss: 0.0264 - val\_loss: 0.0161

This is test result.

pred = model.predict([x\_text[:, 0], x\_text[:, 1]])

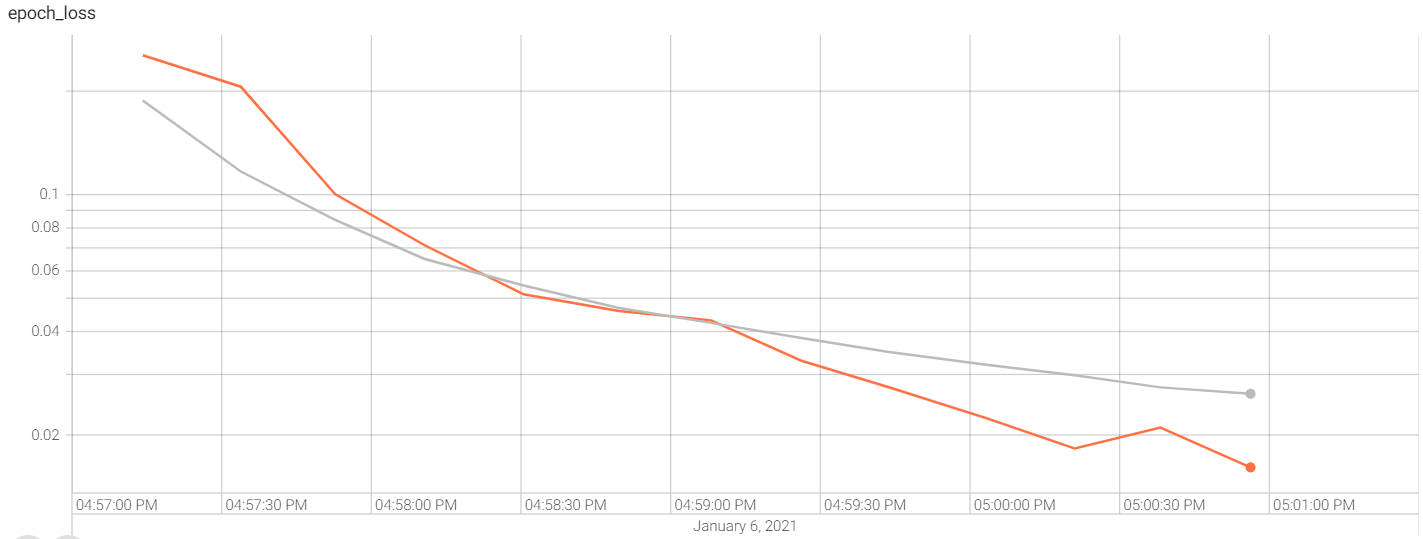
def compute\_accuracy(predictions, labels):

return labels[predictions.ravel() < 0.5].mean()

compute\_accuracy(pred, y\_test)

0.9689655172413794

This is train and validation loss graph



These are examples.

Example1 (same person)

compare\_face(1, 1)

[[10.791455]]

7 6

Example2 (same person)

compare\_face(20, 20)

[[12.081311]]

5 6

Example3 (same person)

compare\_face(40, 40)

[[16.283669]]

4 0

Example4 (different person)

compare\_face(5, 15)

[[109.06947]]

6 0

Example5 (different person)

compare\_face(40, 3)

[[249.01291]]

2 1

Example6 (different person)

compare\_face(40, 41)

[[82.62711]]

7 3

## Audio detection(Dog vs Cat)

### Model description

This is application of Siamese network. The objective of this network is to differ sound of dog and cat. We use FCMM model to extract features from audio. We use 3-layered DNN for feature extraction model. We use contrastive loss for loss function. We use Euclidean distance for energy function.

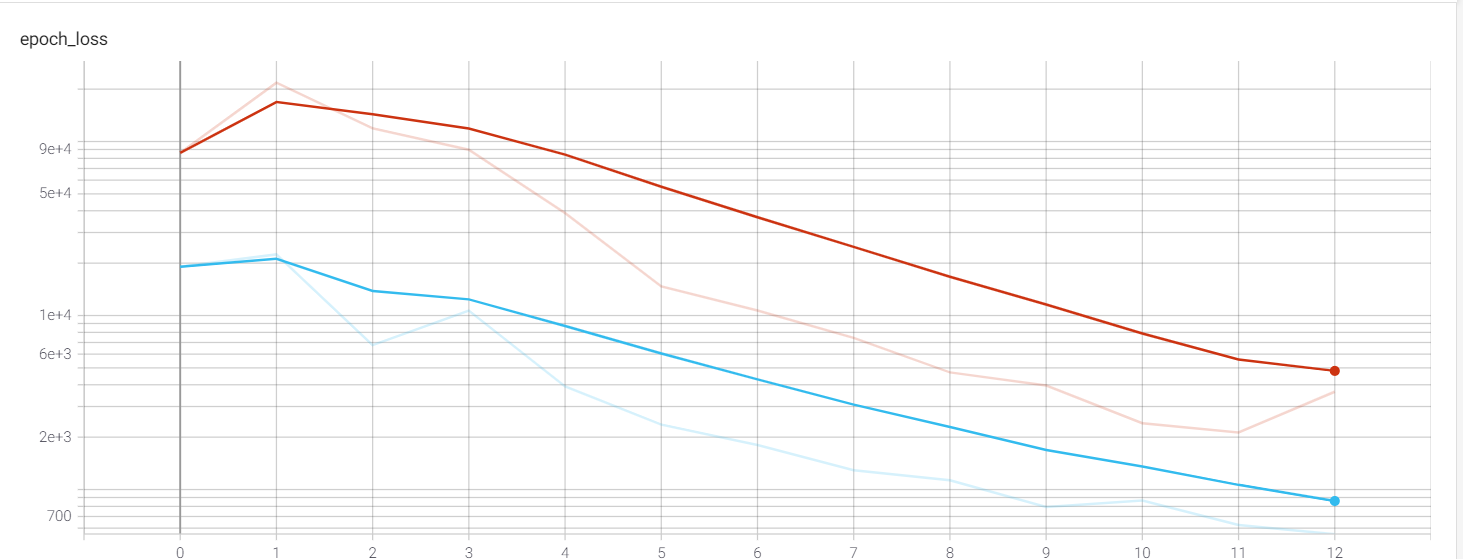
### Data preprocessing

The total datasets compose of 3 audio files of dog, 11 audio files of dog for comparison purpose, 11 audio files of cat for comparison purpose. We make pairs grouping by 3 dog audio files to both of comparison audio files of dog and cat. The total dataset size is 11, which is divided into 8 training data and 3 test data.

### Training Result

This is training results. Training results are composed of train-validation loss graph, data of training loss for each epoch.

This train-val loss graph



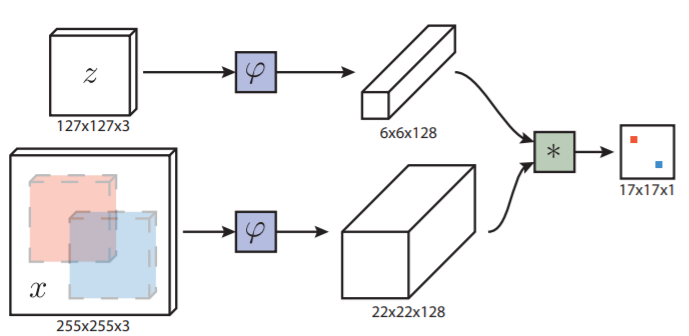
## Fully-Convolutional Siamese Networks for Object Tracking

### 논문 요약

이 논문은 2016년도 10월달에 작성된 논문이다. 초고에서 하는 설명은 기존 object tracking algorithm의 문제점을 소개하고 Fully-Convolutional Siamese Network를 이용해서 object tracking을 해결할 것을 제안한다.

Introduction 부분에서는 왜 Fully-Convolutional Siamese Network를 통해 문제를 해결해야하는지에 대해서 설명한다. 기존 시스템의 문제를 크게 두가지로 보는데 첫번째가 supervised data의 희소성, 두번째가 기존 모델이 단순할 수 밖에 없는 구조가 이에 해당 한다.

이 모델의 핵심 기술인 Siamese Network는 Similarity를 구하는 함수이다. 두개의 물체의 사진을 보고 similarity를 판단하는 것이 이 함수의 목표이다. 이 것을 object tracking에 적용하기 위해서 이 논문에서 제안한 방법은 다음과 같다. 함수의 목표는 하나의 exampler image를 search image 속에서 찾는 것(locating) 이 목표이다. 이를 해결하기 위해서 다음과 같은 구조의 모델을 제안한다. 이 모델은 이미지 크기가 다른 두 인풋을 Siamese network에서 처리하기 위해서 만든 구조이다. Search image속에서 exampler image를 찾기 위해서 search image를 여러 개의 등분으로 나누고 나눈 이미지와 search image의 similarity를 비교하는 방식이다.



여기서 search image를 여러 조각으로 나눌 때 실제로 나누는 것이 아니라 하나의 이미지를 embedding function 속에 넣고 나온 결과를 가지고 cross correlation layer의 인풋으로 대입한다. Cross correlation layer의 핵심은 두개의 인풋 모두 spatial한 정보를 포함하고 있다는 것을 이용해서 아웃풋의 score map의 각 element가 similarity를 나타낸다는 것이다.

f(z, x) = ϕ(z) ∗ ϕ(x) + b 1

이를 나타내는 것이 위의 수식이다. 이 수식이야 말로 이 논문의 핵심이자 크기가 다른 두 사진 속에서 object tracking을 가능케하는 마법 같은 알고리즘의 중심 코드이다.

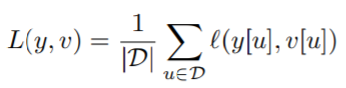
논문의 나머지 부분은 모델의 loss를 구하는 방법과 pair를 구성하는 방법에 대해 설명하고 있다. 여기서는 실제 데이터 셋을 구성하는 방법과 라벨링을 하는 방법 그리고 loss를 정의하는 방법에 대해서 설명하고 있다.



데이터 셋을 구성할 때 이미지 페어는 위의 그림과 같이 전 위치에서의 물체만 크롭한 사진과 그 다음 위치에서의 전체 사진으로 구성되었다. 전체 사진에서 각 위치마다 라베링을 하는 방법은 크롭한 사진의 중심위치로부터의 거리로 정해진다.

또한 loss의 경우 logistic loss의 mean값을 사용하고 optimizer는 Stochastic Gradient Descent를 사용한다.





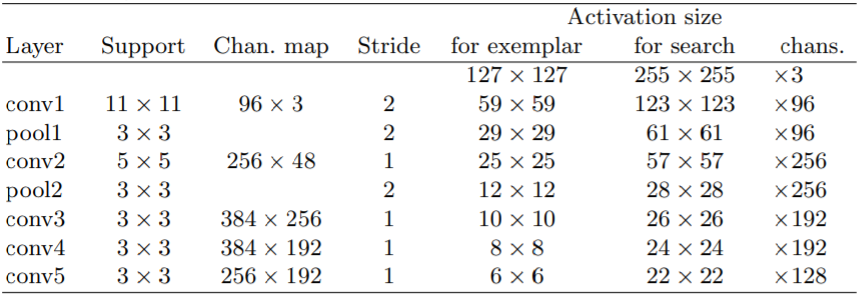


### 데이터셋

During training, we adopt exemplar images that are 127 × 127 and search images that are 255 × 255 pixels. Images are scaled such that the bounding box, plus an added margin for context, has a fixed area. More precisely, if the tight bounding box has size (w, h) and the context margin is p, then the scale factor s is chosen such that the area of the scaled rectangle is equal to a constant s(w + 2p) × s(h + 2p) = A . (7) We use the area of the exemplar images A = 1272 and set the amount of context to be half of the mean dimension p = (w + h)/4. Exemplar and search images for every frame are extracted offline to avoid image resizing during training. In a preliminary version of this work, we adopted a few heuristics to limit the number of frames from which to extract the training data. For the experiments of this paper, instead, we have used all 4417 videos of ImageNet Video, which account for more than 2 million labelled bounding boxes.

### 모델 구조

The architecture that we adopt for the embedding function ϕ resembles the convolutional stage of the network of Krizhevsky et al. [16]. The dimensions of the parameters and activations are given in Table 1. Maxpooling is employed after the first two convolutional layers. ReLU non-linearities follow every convolutional layer except for conv5, the final layer. During training, batch normalization [24] is inserted immediately after every linear layer. The stride of the final representation is eight. An important aspect of the design is that no padding is introduced within the network. Although this is common practice in image classification, it violates the fully-convolutional property of eq. 1



### 연구 결과

### 결론