Artificial Neural Networks & Deep Learning

HW #3

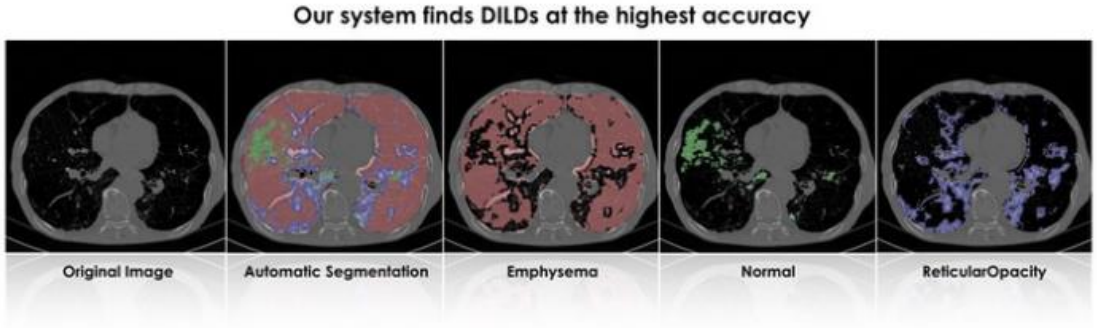
국내기업

VUNO

[비글로벌 스타트업 배틀 #9] 폐암 진단 소프트웨어 스타트업 뷰노 코리아, "의사의 오진을 기술로 극복한다"

EDITOR'S PICK

May 15, 2015 유 재연

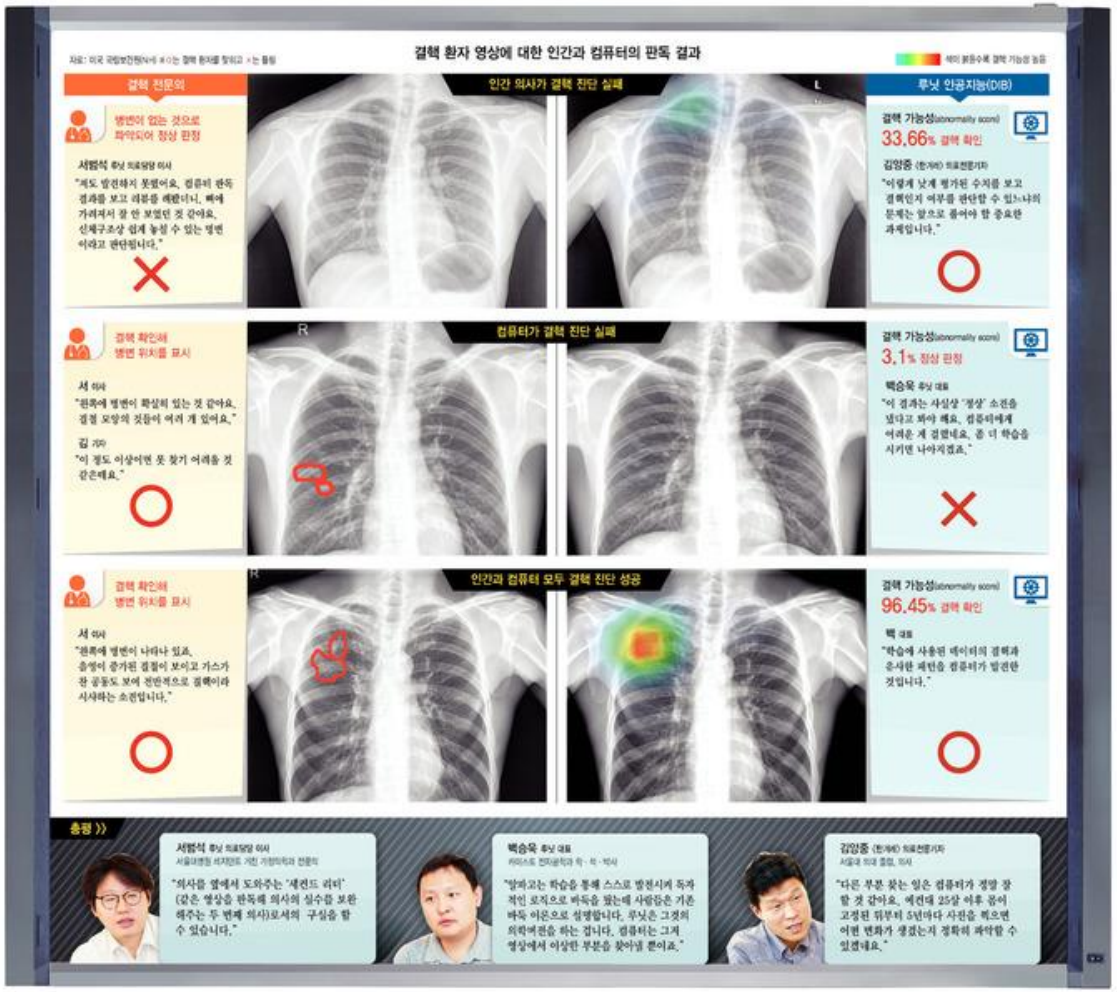


두 명의 의사가 같은 CT 촬영 사진으로 진단할 때 일치할 확률은 60%다. 또한, 미국에서 의사의 오진으로 사망하는 환자의 수는 일년에 약 4만 명이라고 한다. 비싼 의료비와 의사라는 전문직에 대한 신뢰도에 비하면 실망스러울 수 있지만, 의사도 사람이기 때문에 정확하지 않을 수 있다. 문제는 환자의 입장이다. 생사를 오가는 오진을 너그러운 마음으로 받아들일 수 있는 사람은 몇 없다.

https://www.vuno.co/

국내기업

Lunit



https://lunit.io/

http://www.hani.co.kr/arti/science/science_general/755594.html

왜 의료영상 분야를 선택했나요?

의학에서는 단 1%의 성능 향상으로도 매우 많은 사람의 생명을 살릴 수 있습니다.

따라서 AI 기술이 의료에 적용되면, 다른 분야에 비해 더욱 가치 있게 쓰일 수 있습니다. 의료영상을 통해 환자를 진단하는 과정은 상당 부분 컴퓨터 비전(Computer Vision) 문제와 유사합니다. 입력된 영상으로부터 환자의 상태를 최대한 정확하게 추정해야 하기 때문입니다. 병원 내에 저장된 대규모의 영상 및 임상 데이터를 최대한 활용하면, 최근 AI 기술들이 보여주고 있는 놀라운 성과들을 의학에서도 보게 될 것입니다.

또한 의료영상에 AI를 접목하면 새로운 의학적 발견의 가능성이 있습니다. 지금까지의 의학은 전문가의 경험을 통해 특정 패턴을 발견하고 정의하며 발전해왔습니다. 반면 AI가 접목된 의료는 대규모의 영상 및 임상 데이터로부터 전문가가 발견하지 못했던 복잡하고 새로운 패턴을 발견할 수 있습니다.

https://lunit.io/joinus/

Chest X-ray (Pneumonia 폐렴)

- Classification problem
  - input variable: images!
  - 1 binary output variable (pneumonia or normal)
- 5863 x-ray images
  - Already split into train, validation and test.

- https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
- It's already downloaded into the server. You may just copy it into your home.
  - /home/EIEN443/chest_xray.zip

Start Early! (indeed…)

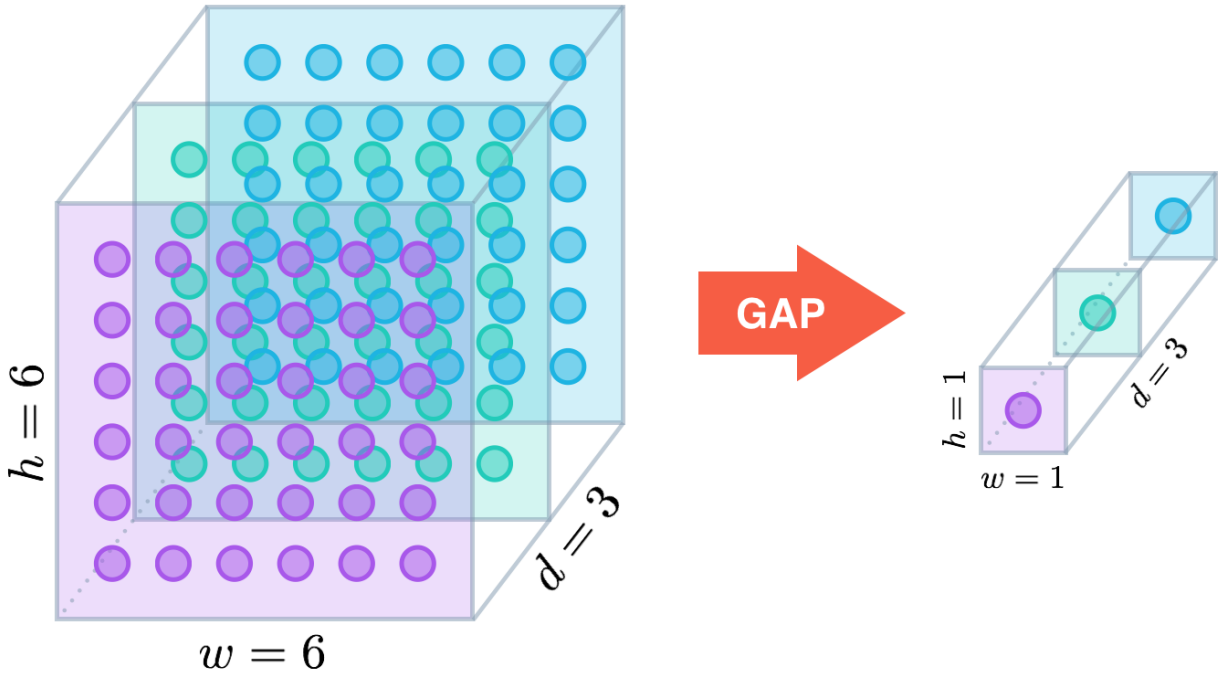
- In this homework, you will do “transfer learning” and it will take quite much time.

- On the GPU server, the estimated time to finish learning is…
  - Q1: 3~4 hours
  - Q2: 3~4 hours
  - Q3: 8~10 hours
  - Q4: 5~6 hours
  - QE1: 1 hour
  - QE2: 1 hour
  - Total: 20~24 hours

- This estimated time is
  - 1) when you do not have any mistake.
  - 2) where there is no other students want to use the server (if all GPUs are in use, you should wait!)
  - 3) when you save all the resulting model properly (read the HW3 template and this document carefully, especially QE2. If you miss something, you should learn the model in the Q1-Q3.)

Global Average Pooling 2D (GAP)

- Each response map is averaged into single neuron.
- Thus, the GAP output of *n* feature maps is *n* neurons regardless the width and heights of feature maps.



Q1: 2 points

- Data preprocessing: The image sizes all vary. Thus, resizing is essential.
  - When loading images, resize the image into [128, 128]
  - flow_from_directory(train_dir, target_size=(128,128), batch_size=20,class_mode='binary')

- Base model: you will use a pre-trained model, VGG16 (weights='imagenet').
- Classifier: the top MLP structure should be:
  - GlobalAveragePooling2D ->
  - Dense(512) -> BatchNormalization -> Activation(Relu) ->
  - Dense(128) -> Dense(1)

- You should do 2-step fine-tuning
  - 100 epochs for the frozen base + 50 fine-tuning epochs (only tune 5-blocks)
  - Learning parameters: RMSprop with learning rate of 1e-5
  - When you load a model, you should set optimizer again.
- You can run multiple times and average the results. However, I do not recommend, since it will take quite long time to learn (more than 2 hours using a GPU)

- Fill the table of the HW3 template.
  - Show your codes, accuracy, and loss in the training and test set,
  - Also show the accuracy graph and loss graph in the training and validation set.

- **Note. Do not forget saving the learned model before and after fine-tuning. You will use the saved model in QE2**

Q2: 1 point

- The previous model showed serious overfitting. Thus, let's add **dropout.**
- The modified classifier: the top MLP structure should be:
  - GlobalAveragePooling2D -> Dropout(0.25) ->
  - Dense(512) -> BatchNormalization -> Activation(Relu) -> Dropout(0.25) ->
  - Dense(128) -> Dropout(0.25) -> Dense(1)
- You should do 2-step fine-tuning
  - 100 epochs for the frozen base + 100 fine-tuning epochs
  - All other parameters should be same with the problem 2's
- Fill the table of the HW3 template.
  - Show accuracy, and loss in the training and test set,
  - Also show the accuracy graph and loss graph in the training and validation set.
- Do you think that overfitting is reduced?
- Is it improved compared to the results of Q1?
- **Note. Do not forget saving the learned model before and after fine-tuning. You will use the saved model in QE2**

Q3: 1 point

- Repeat Q2 with image resizing into [256, 256] and [512, 512].
  - For [512, 512], due to memory limitation, you should change the batch size into 10.
  - For [256, 256], the batch size of 20 is okay. (No change is required)

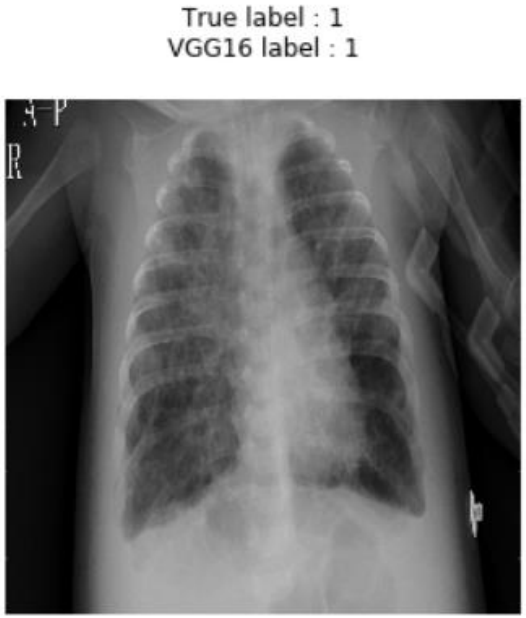
- You should do 2-step fine-tuning
  - All parameters should be same with the problem 3's

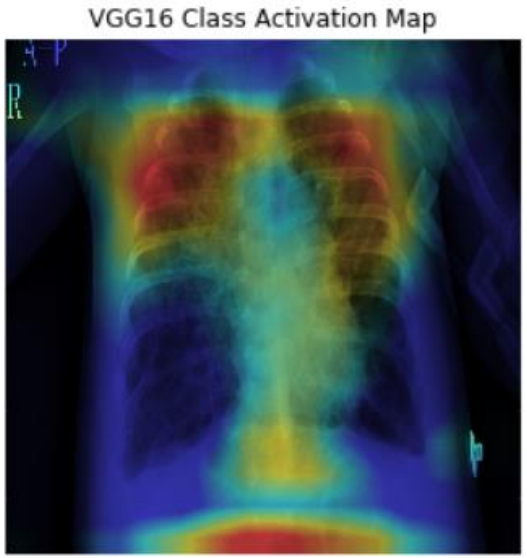
- Fill the table of the HW3 template.
  - Show accuracy, and loss in the training and test set,
  - Also show the accuracy graph and loss graph in the training and validation set.
- Which one is the best among results among Q1 Q2, and Q3? Why?

- **Note. Do not forget saving the learned model before and after fine-tuning. You will use the saved model in the QE2**

Q4: 2 points

- Using the Chapter 5.3 of the textbook, draw the area that was important for classification.
- You can use matplotlib's pyplot.imshow.
- The results should be similar to below.





QE1: Extra 0.5 points

- We will try CNN for varying image sizes.
- For data_flow_directory, do not specify resize. In other words, simply
  - flow_from_directory(train_dir, batch_size=20,class_mode='binary')
- Instead, the input_shape of CNN should be specified as [None, None, 3]
  - input_shape = [None, None, 3]

- You should do 2-step fine-tuning
  - All parameters should be same with the problem 4's

- Fill the table of the HW3 template.
  - Show accuracy, and loss in the training and test set,
  - Also show the accuracy graph and loss graph in the training and validation set.
- Run the code. Does it work?
- Replace GlobalAveragePooling2D with Flatten. Run the code, does it work?
- Does it work better than the best model of Q3? If so, why? If not, why?

- **Note. Do not forget saving the learned model before and after fine-tuning. You will use the saved model in QE2**

QE2: Extra 0.5 points

- There are other methods to evaluate the model.
- Compute the following scores in Q1~Q3 (and QE1).
  - Precision, Recall (sensitivity), Specificity, F1 score, AUC
  - These scores should be computed in the test data set only.

- You need to use sklearn (adapt it into your code!)
  - y_pred=model.predict_generator(test_generator)
  - matrix = sklearn.metrics.confusion_matrix(y_test, y_pred>0.5)
  - auc=sklearn.metrics.roc_auc_score(y_test, y_pred)
- Which model was the best considering all of the computed scores?
- (Recall은 뭐가 좋았고, Precision은 뭐가 좋았고, **이런 식의 답이 아니라,** 여러 스코어를 종합적으로 보았을 때 어떤 모델이 가장 뛰어난지를 비교하라는 뜻임. 혹은 다른 score보다 어떤 score가 중요하므로, 어떤 모델이 좋다라는 결론이던가…)
  - https://bcho.tistory.com/1206

QE3: Extra 1 point

- Let's use different CNN base model, inceptionV3(weights='imagenet').
  - Use the same decision maker part with the models in the main questions

- Following Q1-Q3, and QE1, QE2, find the best model. The model should be tested through
  - 2-step fine-tuning (Q1)
  - Avoiding overfitting (Q2)
  - Investigating whether image resizing affects the performance (Q3 and QE1)
  - Various evaluation methods (QE2)

Deadlines & Submission

- Total scores: 6 points + extra 2 points

- Due: May 10. 2021. 11:59 PM (Monday)
  - No grace period.
  - Be punctual, 1 day delay = 1 point penalty.

- How to submit
  - Use Blackboard's assignment tab.
  - 블랙보드(kulms.korea.ac.kr) -> assignments 탭에서 제출
  - No email submission. (이메일 제출 안 받습니다.)

  - Use the given template file to write a report.
  - 주어진 템플릿 파일 사용해서 리포트 작성.

  - Also submit all of your codes you used. Make a file for each problem (e.x. hw3_Q1.py, hw3_Q2. py, etc). The compressed file name should be **hw3_codes.zip**. If you do not submit the codes, the score will not given.
  - 사용하신 모든 코드를 압축해서 제출해주세요. 문제번호별로 파일을 만들어서 제출해주세요 (예를 들어 hw3_Q1.py, hw3_Q2. py 같이) 압축화일 이름은 **hw3_codes.zip**으로 해주세요. 코드를 제출 안 하시면 숙제 점수는 0점입니다.