Application of Machine Learning to Anomaly Detection in Chest X-Rays

Jacob A. Knox

Department of Mathematics, Florida Southern College

Author Note

Jacob Knox https://orcid.org/0009-0008-1240-9280

I do not have any conflicts of interest to disclose.

Correspondence concerning this article should be addressed to Jacob A. Knox. Email: jacobknoxa@gmail.com.

Abstract

This paper outlines a custom-built convolutional neural network trained on thousands of chest x-ray images to identify a variety of anomalies that may be present in a given image. The roughly 149,000 images used in this paper were gathered from various Kaggle datasets and pre-processed via sorting, resizing, and splitting to create training, validation, and testing datasets with equal distributions of each class to be used with the model. The model itself was then developed in Python using Keras (TensorFlow) with accompanying code to train and test the model. Due to poor coding practices and technical difficulties that ensued, this paper is not able to present the accuracy of the trained model on the training and validation datasets; however, it was determined that the model achieved roughly 37% accuracy on novel testing samples, which is better than randomly guessing but worse than labeling all samples as normal (about 45% of all samples were normal). Although the model falls short of similar research in the domain, it is lightweight and requires a relatively short time to train and predict. Moving forward, there are multiple ways this model can be improved and for research in the domain of medical artificial intelligence models to improve.

All code for this project can be found at this GitHub repository: https://github.com/JacobKnox/Anomaly-Detection-in-Chest-Xrays.

Application of Machine Learning to Anomaly Detection in Chest X-Rays

This paper shall present a convolutional neural network designed to identify sixteen different anomalies—adding the absence of the sixteen anomalies, labeled as normal, brings the total number of classes to seventeen—present in images of chest x-rays from a variety of sources.

Importance of Chest X-Rays

Chest x-rays are a common, relatively inexpensive test that help doctors with diagnosing and monitoring a wide variety of things. For example, Mayo Clinic Staff state that these images can help "determine whether you have heart problems, a collapsed lung, pneumonia, broken ribs, emphysema, cancer or any of several other conditions" as well as monitoring changes after operations, pacemakers, catheters, and defibrillators.

As for cost, according to Healthcare Bluebook, the price of a chest x-ray in a 50 mile radius of Orlando, FL ranges from $68–79 (with an outlier of $137) whereas a chest CT (no contrast) is $243–355 and a chest MRI is $370–620. This geographic area was chosen, because Orlando is the fourth largest city in Florida by population, it hosts two of the top ten hospitals by staffed beds in Florida, it is close to Lakeland while still ensuring the majority of the radius is over land, and 50 miles was the maximum radius Healthcare Bluebook would allow. It was determined that these factors would make this geographic area a fair representation of prices across the state without performing a multitude of queries.

Deep Learning Approaches to Solving Problems

There are a wide variety of deep learning algorithms that are each generally useful for solving problems in specific domains. Since this paper's goal involves identifying something

within images, its domain falls under computer vision, which is generally considered to be the speciality of convolutional neural networks. The next two paragraphs shall explain what neural networks and convolutional neural networks are for readers who may not be fully familiar with the subject; however, the reader can assume that this paper may not attempt to elaborate on all concepts or terms mentioned for the sake of brevity.

Neural networks are networks of interconnected nodes that attempt to loosely model the human brain and its functionality. They consist of multiple node layers including an input layer, one or more hidden layers (widely considered to be a "black box"), and an output layer. IBM explains well how these nodes interact:

> Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

As designers of these networks, we control the number of nodes in each layer, the number of layers, and a handful of design parameters such as the threshold value. From there, the network will learn the associated weights and, if designed well, what is and is not important.

Convolutional neural networks are neural networks that have three primary types of layers: convolutional, pooling, and fully-connected layers. Convolutional layers extract important features from the input by passing multiple weighted filters over the image to create multiple feature maps. These feature maps then generally go through a pooling layer where another filter is passed over them, but this filter performs data reduction by generally either returning the max or average value inside the filter. This process is repeated multiple times to

reduce the data's size significantly before passing it to a fully-connected layer. Fully-connected layers are layers where every node of the layer is connected to every node of the previous layer. This is generally the layer directly before the output layer in a convolutional neural network. The output layer then generally changes depending on the application: it will often either be one node or n nodes where n is the number of classes the model is meant to predict. If there is only one node, it can either represent a binary classification or a probability. If there are multiple nodes, it can either be probabilities of each class or multiple labels (i.e.: a sample belongs to two classes, so the model should activate the two corresponding nodes).

Background Research

Research surrounding applications of convolutional neural networks, among other artificial intelligence techniques, to medical imaging and diagnosis have been on the rise, especially during the peak of the COVID-19 pandemic. The following research is only approximately half of the COVID-19 specific papers found during background research conducted: Samir et al., 2023; Wang & Hargreaves, 2022; Appasami & Nickolas, 2022; Khan et al., 2022; Sadre et al., 2021; Shelke et al., 2021; Karhan & Akal, 2020; Yoo et al., 2020. Some of the papers focused on evaluating existing research and how it could be improved (Wang; Sadre) while the others took variable approaches to identifying COVID-19 in chest x-rays: deep convolutional neural networks (Appasami), applying pre-trained models (Khan), binary decision trees (Yoo), or applying existing architectures such as variants of ResNet or DenseNet (Samir; Shelke; Karhan).

Unlike the papers discussed above, this paper presents a fully convolutional network that detects multiple anomalies rather than just one or two. Research focused on the same objective

shall be discussed in two sections: one discussing research similar in scope to this paper and one discussing research beyond the scope of this paper.

Lighter, Similar Approaches

In 2019, Baltruschat et al. published "Comparison of Deep Learning Approaches for Multi-Label Chest X-Ray Classification" seeking to test the powerful ResNet architecture on chest x-ray classification. Ultimately, they tested and compared ResNet-38, ResNet-50, ResNet-101, a network made from scratch, and models taking non-image patient data into consideration. They found that a ResNet-38 architecture taking non-image patient data into consideration had the highest overall accuracy. This paper does not take non-image data into consideration, but does test existing architectures and models against the one presented in this paper.

In 2022, Pillai published "Multi-Label Chest X-Ray Classification via Deep Learning" seeking to develop a lightweight deep learning solution to detect 14 anomalies in chest x-rays. They built a custom convolutional neural network to compare against DenseNet121, ResNet-50, Inception_V3, and VGG16—much like the objective of this paper. They found that the DenseNet121 performed the best, but that each model had shortcomings on detection of specific anomalies which they hypothesized was a result of unbalanced data.

More In-Depth, Advanced Approaches

In 2019, Chen et al. published "Deep Hierarchical Multi-label Classification of Chest X-ray Images" seeking to improve upon existing works surrounding Hierarchical Multi-Label Classification (HMLC) models by considering oncological hierarchies, conditional probability, and unconditional probability. In essence, they considered the overall probability of an

abnormality, the fact that some abnormalities (i.e.: COPD) are beneath others (i.e.: Pulmonary in the case of COPD) in an oncological hierarchy, and the conditional probability of an abnormality given its parent in the hierarchy (i.e.: P(COPD | Pulmonary)). Their HMLC model, taking all of these into consideration, outperformed existing models at the time.

In 2021, Seah et al. published "Effect of a comprehensive deep-learning model on the accuracy of chest x-ray interpretation by radiologists: a retrospective, multireader multicase study" seeking to prove that radiologists' classification accuracy is improved when they have the assistance of a comprehensive deep learning model. They had 20 thoracic radiologists (radiologists who are fellowship trained and have > 5 years of post-training experience) review cases, first without any assistance and then 3 months later with the assistance of a trained deep learning model, to see if the deep learning model significantly improved their performance. Their findings were:

> Unassisted radiologists had a macroaveraged AUC of 0.713 (95% CI 0.645–0.785) across the 127 clinical findings, compared with 0.808 (0.763–0.839) when assisted by the model. The deep-learning model statistically significantly improved the classification accuracy of radiologists for 102 (80%) of 127 clinical findings, was statistically non-inferior for 19 (15%) findings, and no findings showed a decrease in accuracy when radiologists used the deep learning model. Unassisted radiologists had a macroaveraged mean AUC of 0.713 (0.645–0.785) across all findings, compared with 0.957 (0.954–0.959) for the model alone. Model classification alone was significantly more accurate than unassisted radiologists for 117 (94%) of 124 clinical findings predicted by the model and was non-inferior to unassisted radiologists for all other clinical findings.

In short, the deep learning model outperformed the radiologists in 94% of findings and radiologists assisted by the model outperformed their unassisted peers in 80% of findings. Their deep learning model used three different convolutional neural networks with the EfficientNet and U-net architectures.

Methodology

Data Collection and Manipulation

Images were gathered from nine different Kaggle datasets and combined into one massive dataset. They were then sorted into folders based on their respective anomalies with images containing multiple anomalies being duplicated (one for each anomaly). Datasets where images were pre-categorized were simply moved into their respective folders whereas custom sorting scripts (see Appendices A, B, and C) were written in Python to sort datasets that had a comma-separated values (CSV) file or similar means of documenting the anomalies present in each image.

Once the images were successfully sorted, a splitter script (see Appendix D) was written in Python to randomly split each anomaly into roughly 10% validation data, 30% testing data, and 60% training data. Once successfully split, a documenter script (see Appendix E) was written in Python to create a CSV file detailing the image, anomaly, and split category.

Throughout the process of working on a fully connected network, multiple issues were encountered when attempting to work with images of variable sizes including data being too large to fit in memory, Python libraries I frequently use not adequately supporting arrays with variable dimensionality, and difficulties designing the model architecture to accommodate variable input sizes. Because of these difficulties, it was decided that it would be best if the data

were all converted to one consistent size. As such, a resizer script (see Appendix F) was written in Python to resize all images in the dataset to 256 X 256. Images were not cropped nor was aspect ratio preserved, so this may have affected training and testing of the model.

Architecture of Model

The convolutional neural network currently consists of 30 layers with a combined total of 835,091 trainable parameters—see Appendix G for the full model summary. These 30 layers are as follows where each layer is connected to the previous one (including the last layer of one group being connected to the first layer of the next group):

- One input layer with a shape of (None, 256, 256, 1) where the None represents that it can take in any number of images at a time, the next two numbers (256, 256) represent the size of a single input image and 1 indicates it is only one color channel (black and white images).
- Five layer groups of Conv2D–Dropout–BatchNormalization–Activation each using a kernel of (2, 2), a stride of (2, 2), and a ReLU activation. Each group doubles the number of filters from a default of 32 to 512.
- One fully connected layer group of Conv2D–Dropout–BatchNormalization–Activation using a default filter of 64, a stride and kernel of (2, 2), and a ReLU activation.
- One output layer group of Conv2D–Dropout–BatchNormalization–GlobalMaxPooling2D–Activation using a filter of 17, a stride and kernel of (2, 2), and a softmax activation.

Results

The convolutional neural network outlined above, and specifically designed for this paper, achieved roughly 37% accuracy on testing data—due to a variety of circumstances, I was unable to ascertain the model's accuracy on training and validation data. Generally, the goal should be to achieve high accuracy across all datasets mentioned (training, testing, and optionally validation), because this means that the model has learned well from what it has seen and can also generalize well to what it has not seen before. High accuracy on training data but low accuracy on testing data is often an indicator that the model has fallen victim to overfitting on the training dataset meaning that it has confined itself too much—almost no better than a series of nested conditional statements. Since I do not have the training and validation accuracy, I cannot determine the true cause behind the low testing accuracy whether it be overfitting, poor model architecture, or issues with the data itself and the manipulations conducted on it.

This accuracy is, of course, disappointing and much lower than accuracies of other research in the other area. Such an accuracy is not conducive for a model that could be included in the medical field, but with tweaks and further time investment the model could improve greatly.

Conclusion

Work Conducted

Overall, work conducted during the span of this research project includes gathering and processing a large dataset consisting of roughly 149,000 images of chest x-rays from multiple sources for use in an artificial intelligence application, designing and implementing the architecture of a custom-made convolutional neural network, and applying said convolutional neural network to the aforementioned dataset in order for the model to identify sixteen different

anomalies in images of chest x-rays. The accuracy of this applied model on novel testing data, although low, proved that the model works and at least achieves accuracy better than randomly guessing.

Future Work

Dr. Hoan Ngo, Assistant Professor of Computer Science at Florida Southern College, and I have been arranging a directed study for the Spring 2024 semester where a small group of students at Florida Southern College, myself included, will be working on applications of artificial intelligence to the medical field. The idea is that this will facilitate students' learning in the field of artificial intelligence and challenge them to learn more—different approaches, including some that may not have been taught in the concentration offered at the College, and working with PyTorch instead of TensorFlow—while working towards building a network of models that could potentially cooperate to assist doctors in medical diagnoses.

On a broader scale, multimodal—meaning that it accepts a variety of input types such as images, text, numbers, etc.—artificial intelligence models are a promising new area emerging in the medical field. Just earlier this semester in August 2023, Greg Corrado, Head of Health AI at Google Research, and Yossi Matias, Vice President of Engineering and Research at Google Research, published a blog post outlining multiple methods of creating multimodal large language models—models that generally understand and generate language such as ChatGPT or Bard—and introducing their Med-PaLM M model. Med-PaLM M is a multimodal version of their Med-PaLM model, which is in turn a medically-specialized variant of their PaLM-E model designed to answer medical questions. Where these types of models will go and how they will improve the medical world is yet to be seen, but is an exciting new realm of research.

References

Appasami, G., Nickolas, S (2022). A deep learning-based COVID-19 classification from chest X-ray image: case study. *Eur. Phys. J. Spec. Top.* 231, 3767–3777. https://doi.org/10.1140/epjs/s11734-022-00647-x

Baltruschat, I. M., Nickisch, H., Grass, M., Knopp, T., & Saalbach, A. (2019, April 23). *Comparison of deep learning approaches for multi-label chest X-ray classification*. Nature News. https://www.nature.com/articles/s41598-019-42294-8

Chen, H., Miao, S., Xu, D., Hager, G. D., & Harrison, A. P. (2019). *Deep hierarchical multi-label classification of chest X-ray images*. International Conference on Medical Imaging with Deep Learning. https://scholar.archive.org/work/6na35tcdpnd5zbh6gllzdn6hmy

Corrado, G., & Matias, Y. (2023, August 3). *Multimodal Medical AI*. Google Research Blog. https://blog.research.google/2023/08/multimodal-medical-ai.html

Healthcare Bluebook. (n.d.). *Healthcare Bluebook*. Healthcare bluebook – your guide to fair pricing for healthcare services. https://www.healthcarebluebook.com/ui/home

IBM. (n.d.). *What are convolutional neural networks?* https://www.ibm.com/topics/convolutional-neural-networks

Karhan, Z., & Akal, F. (2020, December 25). Covid-19 Classification Using Deep Learning in Chest X-Ray Images. IEEE Xplore. https://ieeexplore.ieee.org/document/9299315

Khan, E., Rehman, M. Z. U., Ahmed, F., Alfouzan, F. A., Alzahrani, N. M., & Ahmad, J. (2022). Chest X-ray Classification for the Detection of COVID-19 Using Deep Learning Techniques. *Sensors*, *22*(3), 1211. https://doi.org/10.3390/s22031211

Mayo Clinic Staff. (2022, March 5). *Chest X-rays*. Mayo Clinic. https://www.mayoclinic.org/tests-procedures/chest-x-rays/about/pac-20393494

Pillai, A.S. (2022) Multi-Label Chest X-Ray Classification via Deep Learning. Journal of Intelligent Learning Systems and Applications, 14, 43-56. https://doi.org/10.4236/jilsa.2022.144004

Sadre, R., Sundaram, B., Majumdar, S. *et al* (2021). Validating deep learning inference during chest X-ray classification for COVID-19 screening. *Sci Rep* 11, 16075. https://doi.org/10.1038/s41598-021-95561-y

Samir, B., Mwanahija, S., Soumia, B., & Özkaya, U. (2023, January 6). *Deep learning for classification of chest X-Ray Images (covid 19)*. arXiv.org. https://arxiv.org/abs/2301.02468

Seah, J. C. Y., Tang, C. M. H., Buchlak, Q. D., et al. (2021, August). *Effect of a comprehensive deep-learning model on the accuracy of chest x-ray interpretation by radiologists: a retrospective, multireader multicase study*. The Lancet Digital Health. https://www.thelancet.com/journals/landig/article/PIIS2589-7500(21)00106-0/fulltext

Shelke, A., Inamdar, M., Shah, V., Tiwari, A., Hussain, A., Chafekar, T., & Mehendale, N. (2021). Chest X-ray Classification Using Deep Learning for Automated COVID-19 Screening. *SN computer science*, *2*(4), 300. https://doi.org/10.1007/s42979-021-00695-5

Shelhamer, E., Long, J., & Darrell, T. (2016, May 20). *Fully convolutional networks for semantic segmentation*. arXiv.org. https://doi.org/10.48550/arXiv.1605.06211

Wang, Y., & Hargreaves, C. A. (2022, November). *A Review Study of the Deep Learning Techniques used for the Classification of Chest Radiological Images for COVID-19 Diagnosis*. ScienceDirect. https://doi.org/10.1016/j.jjimei.2022.100100

Yoo, S. H., Geng, H., Chiu, T. L., Yu, S. K., Cho, D. C., Heo, J., Choi, M. S., Choi, I. H., Cung Van, C., Nhung, N. V., Min, B. J., & Lee, H. (2020, July 14). *Deep learning-based decision-tree classifier for covid-19 diagnosis from chest X-ray imaging*. Frontiers. https://doi.org/10.3389/fmed.2020.00427

Appendix A

Sorter Script for COVIDx CXR-4 Dataset by Zhao et al.

```
import shutil

import numpy as np

import os

ROOT = os.path.dirname(os.path.abspath(__file__))

DATA_ROOT = os.path.dirname(ROOT)

meta = np.loadtxt(f'{ROOT}/train.txt',str,delimiter=" ")

for line in meta:

    if(line[2] == 'positive'):

        src = f'{ROOT}/train/{line[1]}'

        dest = f'{DATA_ROOT}/COVID_19/{line[1]}'

        if os.path.exists(src):

            shutil.move(src, dest)

    else:

        src = f'{ROOT}/train/{line[1]}'

        dest = f'{DATA_ROOT}/TRASH/{line[1]}'

        if os.path.exists(src):

            shutil.move(src, dest)
```

Appendix B

Sorter Script for Chest Xray Masks and Labels Dataset by Pandey

```
import shutil, numpy as np, os
from glob import glob
ROOT = os.path.dirname(os.path.abspath(__file__))
DATA_ROOT = os.path.dirname(ROOT)
options = {"atelectasis": "", "cardiomegaly": "", "consolidation": "",
      "covid-19": "COVID_19", "edema": "", "effusion": "", "emphysema": "",
      "fibrosis": "", "hernia": "", "infiltration": "", "mass": "", "nodule":
      "", "normal": "", "pleural thickening": "PLEURAL_THICKENING",
      "pneumonia": "", "pneumothorax": "", "tuberculosis": "", "tb":
      "TUBERCULOSIS"}
for reading, image in zip(glob('*',root_dir=f'{ROOT}/ClinicalReadings/'),
      glob('*',root_dir=f'{ROOT}/CXR_png/')):
    if reading.split('.')[0] != image.split('.')[0]:
        break
    with open(f'{ROOT}/ClinicalReadings/{reading}', 'r', newline='\n') as file:
        reading_data = file.read().lower()
    for key, value in options.items():
        if key in reading_data:
            src = f'{ROOT}/CXR_png/{image}'
            dest = f'{DATA_ROOT}/{key.upper() if value == "" else
      value}/{image}'
            shutil.copy(src, dest)
```

Appendix C

Sorter Script for NIH Chest X-rays Dataset

```
import numpy as np, shutil, os

ROOT = os.path.dirname(os.path.abspath(__file__))

DATA_ROOT = os.path.dirname(ROOT)

converters = {'No Finding': "NORMAL"}

data = np.loadtxt(f'{ROOT}/Data_Entry_2017.csv', dtype = str, delimiter=",",
      skiprows=1)

for line in data:
    src = f'{ROOT}/images/{line[0]}'
    findings = line[1].split('|')
    for finding in findings:
        dest = f'{DATA_ROOT}/{finding.upper() if finding not in
      converters.keys() else converters[finding]}/{line[0]}'
        shutil.copy(src, dest)
```

Appendix D

Splitter Script to Divide Dataset

```
import shutil, numpy as np, os
from glob import glob
from tqdm import tqdm
ROOT = os.path.dirname(os.path.abspath(__file__))
folders = ['ATELECTASIS', 'CARDIOMEGALY', 'CONSOLIDATION', 'COVID_19', 'EDEMA',
      'EFFUSION', 'EMPHYSEMA', 'FIBROSIS', 'HERNIA', 'INFILTRATION', 'MASS',
      'NODULE', 'NORMAL', 'PLEURAL_THICKENING', 'PNEUMONIA', 'PNEUMOTHORAX',
      'TUBERCULOSIS']
categories = ['TRAIN', 'TEST', 'VALIDATION']
percentages = [1, 0.4, 0.1]
for folder in folders:
    for category in categories:
        os.makedirs(os.path.dirname(f'{ROOT}/{folder}/{category}/'),
      exist_ok=True)
    for image in tqdm(glob('*.png', root_dir=f'{ROOT}/{folder}/'),
      desc=f'Splitting images in folder {folder}', unit="img"):
        src = f'{ROOT}/{folder}/{image}'
        category = categories[0]
        random_num = np.random.rand()
        if(random_num < percentages[2]):
            category = categories[2]
        elif(random_num < percentages[1]):
            category = categories[1]
        dest = f'{ROOT}/{folder}/{category}/{image}'
        shutil.move(src, dest)
```

Appendix E

Documenter Script to Create CSV File

```
from glob import glob
import os, csv
from tqdm import tqdm
ROOT = os.path.dirname(os.path.abspath(__file__))
folders = ['ATELECTASIS', 'CARDIOMEGALY', 'CONSOLIDATION', 'COVID_19', 'EDEMA',
      'EFFUSION', 'EMPHYSEMA', 'FIBROSIS', 'HERNIA', 'INFILTRATION', 'MASS',
      'NODULE', 'NORMAL', 'PLEURAL_THICKENING', 'PNEUMONIA', 'PNEUMOTHORAX',
      'TUBERCULOSIS']
categories = ['TRAIN', 'TEST', 'VALIDATION']
with open(f'{ROOT}/image_docs.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    field = ["image", "anomaly", "split category"]
    writer.writerow(field)
    for folder in folders:
        for category in categories:
            for image in tqdm(glob("*",
      root_dir=f'{ROOT}/{folder}/{category}'), desc=f'Documenting files in
      {ROOT}/{folder}/{category}'):
                writer.writerow([image, folder, category])
```

Appendix F

Resizer Script for Images

```
import numpy as np
from PIL import Image, UnidentifiedImageError
from tqdm import tqdm
DATA_DIR = "C:\\Users\\epicd\\Documents\\Data"
INFO_DIR =
"C:\\Users\\epicd\\Documents\\GitHub\\Anomaly-Detection-in-Chest-Xrays\\image_d
ocs.csv"
def main():
    data_info = np.loadtxt(INFO_DIR, dtype=str, delimiter=",", skiprows=1)
    for row in tqdm(data_info[126000:], desc="Resizing images", unit="image"):
        file_loc = f"{DATA_DIR}\\{row[1]}\\{row[2]}\\{row[0]}"
        try:
            im = Image.open(file_loc)
        except UnidentifiedImageError:
            print(f"Couldn't load image {file_loc}.")
            continue
        out = im.resize((256, 256))
        try:
            out.save(file_loc)
        except OSError:
            print(f"Failed to resize and save image: {file_loc}.")
            continue
if __name__ == "__main__":
    main()
```

Appendix G

Convolutional Neural Network Model Summary

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 256, 256, 1)]	0
conv2d (Conv2D)	(None, 128, 128, 32)	160
dropout (Dropout)	(None, 128, 128, 32)	0
batch_normalization (BatchNormalization)	(None, 128, 128, 32)	128
activation (Activation)	(None, 128, 128, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	8256
dropout_1 (Dropout)	(None, 64, 64, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 64, 64, 64)	256
activation_1 (Activation)	(None, 64, 64, 64)	0
conv2d_2 (Conv2D)	(None, 32, 32, 128)	32896
dropout_2 (Dropout)	(None, 32, 32, 128)	0
batch_normalization_2 (BatchNormalization)	(None, 32, 32, 128)	512
activation_2 (Activation)	(None, 32, 32, 128)	0
conv2d_3 (Conv2D)	(None, 16, 16, 256)	131328
dropout_3 (Dropout)	(None, 16, 16, 256)	0
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 256)	1024
activation_3 (Activation)	(None, 16, 16, 256)	0

Appendix G (cont.)

conv2d_4 (Conv2D)	(None, 8, 8, 512)	524800
dropout_4 (Dropout)	(None, 8, 8, 512)	0
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 512)	2048
activation_4 (Activation)	(None, 8, 8, 512)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	131136
dropout_5 (Dropout)	(None, 4, 4, 64)	0
batch_normalization_5 (BatchNormalization)	(None, 4, 4, 64)	256
activation_5 (Activation)	(None, 4, 4, 64)	0
conv2d_6 (Conv2D)	(None, 2, 2, 17)	4369
dropout_6 (Dropout)	(None, 2, 2, 17)	0
batch_normalization_6 (BatchNormalization)	(None, 2, 2, 17)	68
global_max_pooling2d (GlobalMaxPooling2D)	(None, 17)	0
activation_6 (Activation)	(None, 17)	0

==========

Total params: 837237 (3.19 MB)

Trainable params: 835091 (3.19 MB)

Non-trainable params: 2146 (8.38 KB)

_____

None

Total number of layers: 30