2021 年臺灣國際科學展覽會
優勝作品專輯

作品編號　**090019**

參展科別　醫學與健康科學

作品名稱　**Consensus-based Machine Learning Model in the Scoring for Hepatic Steatosis Severity**

得獎獎項　大會獎 四等獎

就讀學校　國立中科實驗高級中學

臺中市立臺中女子高級中等學校

臺中市立臺中第一高級中等學校

指導教師　陳思佑、趙偉廷

作者姓名　李亭寬、張倍語、陳昀浚

關鍵詞　**artificial intelligence、fatty liver、pathology**

作者簡介

We are Ting-Kuan Lee, Bei-Yu Chang, and Yun-Chun Chen. We were junior high school classmates, but went to different high schools (Taichung Municipal Taichung Girls Senior High School, National Experimental High School at Central Taiwan Science Park, and Taichung Municipal Taichung First Senior High School, respectively) after we graduated. Even still, our passion for exploring the world of science brought us back together and formed the team we have today. Even though our interests, which are respectively computer science and medicine, initially clashed, but from this conflict produced an innovative idea which is now the topic of our research. We deeply appreciate all the people who gave us assistance along the way of our study; we couldn’ t have done it without them.

Abstract

Aims: The prevalence of fatty liver is rising in modern countries worldwide. However, the scoring for fatty liver severity as evaluated by different pathologists may be inconsistent, and a helpful reference opinion is usually lacking. The shortage of pathologists in comparatively rural areas is another issue. We therefore aimed to build a histopathological Artificial Intelligence (AI) model for evaluating the severity of hepatic steatosis.

Methods: With the approval of the Institutional Review Board of Taichung Veterans General Hospital, we screened the pathology database to obtain a sufficient number of liver pathology slices. Two pathologists independently interpreted the percentage of hepatic steatosis (Class 0-3) in each slide. A scoring consensus was reached via a discussion regarding the slides, with different scores given between the two pathologists. Based upon the consensus scores, we trained a machine learning model with the machine learning program, Google Teachable Machine, and optimized its performance after adequate retraining.

Results: Trained using a total of 200 slides, a consensus-based AI model was built, and the model's accuracy growth curve eventually reached a stable performance of 0.8. We performed an independent test with another 100 slides for the two pathologists, and the consensus answers were finally obtained after discussing for the 12 different interpretations. Furthermore, in the independent test that had been interpreted by the two pathologists, the predictions from the AI model were comparable with the consensus scores (71% in the same classes, 27% in the classes with one-class interval, only 2% in the classes with 2-class intervals, and zero percent in the classes with > 2-class intervals).

Conclusions: This consensus-based machine learning model was able to primitively evaluate the severity of hepatic steatosis, and may provide a solid basis for clinical applications in the future.

摘要

研究目的：脂肪肝的盛行率在全世界的現代化國家都在增加。病理醫師在評估脂肪肝的嚴重等級時，常缺乏參考意見以減少差異。許多地區也缺乏病理醫師。本研究旨在建立一個有效評估脂肪肝嚴重程度的病理組織學人工智慧模型。**研究過程：**本研究經由臺中榮民總醫院人體研究倫理審查委員會審核通過，篩選後取得病理資料庫中適合的肝臟組織切片，由兩位病理醫師獨立為肝組織中脂肪堆積的程度評分。再以病理醫師討論後的共識答案為分級標準，來訓練人工智慧模型。**研究結果：**在 100 個樣本的獨立測試中，人工智慧模型和病理專科醫師的評分，有 71%完全相同、27%差異只有一個等級、2%差異 2 個等級、而沒有 2 個等級以上的差異。**結論及應用：**我們已初步建立一個可以評估脂肪肝嚴重程度的人工智慧模型。這模型可為將來人工智慧的臨床應用，建立一個良好的基礎。

Introduction

Liver is an essential organ for fat metabolism, and so-called “fatty liver” may occur if the metabolism is unbalanced (Chalasani, Younossi et al. 2018). Fatty liver is pathologically a short term for the phenomena of fat over-accumulation within hepatocytes. It’s generally the consequences of excessive alcohol intake or fat-related metabolic disorders; for example, being overweight (Cusi 2012). Fatty liver is essentially a lifestyle disease. In Westernized countries, owing to the trend of exquisite diets with a high calorie content, a decreasing amount of exercise, and the popularity of alcohol consumption, fatty liver has become a common disease. Furthermore, an aging society results in the increased risk of fatty liver disease, involving metabolic syndromes such as hypertension, hyperglycemia, and hyperlipidemia (Chalasani, Younossi et al. 2012). Therefore, the rising prevalence of fatty liver disease has been observed throughout modern countries worldwide (Li, Zou et al. 2019). For example, in an epidemiological survey of 3,260 participants launched in Shengang Township, Changhua County, Taiwan, amongst all participants with abnormal liver function, up to 33.6% of cases were caused by fatty liver disease (Chen, Huang et al. 2007).

Fatty liver is just a general term for a spectrum of this disease (Satapathy and Sanyal 2015). As it varies with the progression of the illness, clinical manifestations of fatty liver are differentiated into several stages according to the severity of patients’ condition. The first stage of fatty liver is hepatic steatosis, which doesn’t cause inflammation, though fat particles are piling up in the liver. The second stage is steatohepatitis, where in addition to steatosis, there will be inflamed cells as well as hepatocyte ballooning. In the long term, this could cause liver fibrosis. The final stage is drawn out by chronic liver inflammation, necrosis, and fibrosis, which beget liver cirrhosis. Severe liver cirrhosis is the consequence of liver fibrosis, and may trigger complications, cause liver cancer, or even result in death (Lee, Wu et al. 2017). In an epidemiological research study (Sheka, Adeyi et al. 2020), after keeping track of the participants for roughly a decade, 20% of those who

originally settled in the first stage, steatosis, had progressed to the second stage, steatohepatitis. Moreover, 20% of those who started in the second stage ended up with liver cirrhosis. In a study from the United States (Younossi, Otgonsuren et al. 2015), patients with liver cancer led by fatty liver increased by 9% per year. It is estimated that fatty liver may become the main cause of liver cancer. In the near future, fatty liver is certainly going to become one of the most concerning topics of public health worldwide.

Over the recent decades, Artificial Intelligence (AI) has gone through numerous evolutions. As the technology matures, the fields of its practice continue to extend further and deeper. It can provide great support in biotechnology, and even advance its clinical applications (Le Berre, Sandborn et al. 2020). For example, AI has been used for interpreting cancerous lesions during a gastrointestinal endoscopic examination, detecting bleeding during a capsular endoscopy, evaluating the degree of liver fibrosis, and distinguishing pancreatitis from pancreatic cancer. Furthermore, AI may even assist in the speculation of clinical therapeutics. In the pathological image of fatty liver disease, liver cells with fat accumulation are significantly different when compared to normal liver cells (Kleiner, Brunt et al. 2005). The white round lesions seen in the images are clearly distinguishable from normal liver cells, and are therefore suitable for use in image recognition technology. Diagnosis of fatty liver requires well trained and experienced pathologists; however, the shortage of pathologists in less developed areas is currently a very relevant issue. The areas lacking proper medical resources must send their liver specimens to medical centers simply to get an interpretation. AI can possibly offer an elementary interpretation and provide a prompt diagnostic reference prior to obtaining an interpretation from a pathologist. Additionally, the decisions made by different pathologists may also be varied. It is at this time when AI technology can come into play, as it can serve as a helpful second opinion for pathologists prior to making their final decisions.

For now, the research surrounding AI in the diagnosis of fatty liver disease is still limited. In a recent AI study (Forlano, Mullish et al. 2020), although the team was able to correctly calculate the

percentage of hepatic steatosis in liver tissues, there seemed to be a considerable difference between the technically "accurate" result and a pathologist's classification standard. Therefore, previous research has not been very effective in the clinical classification of fatty liver. To find a solution to the above-mentioned concerns, we aimed to build a machine learning model for scoring the severity of hepatic steatosis based upon pathologists' interpretations.

Materials and Methods

Database screening

Figure 1 shows the process regarding the building of a machine learning AI model. We screened the Pathology Database of Taichung Veterans General Hospital, with patients who had received liver surgery or liver biopsy being recruited for data collection. This study has been approved by the Human Research Ethics Review Committee of Taichung Veterans General Hospital (No. CE20345B).

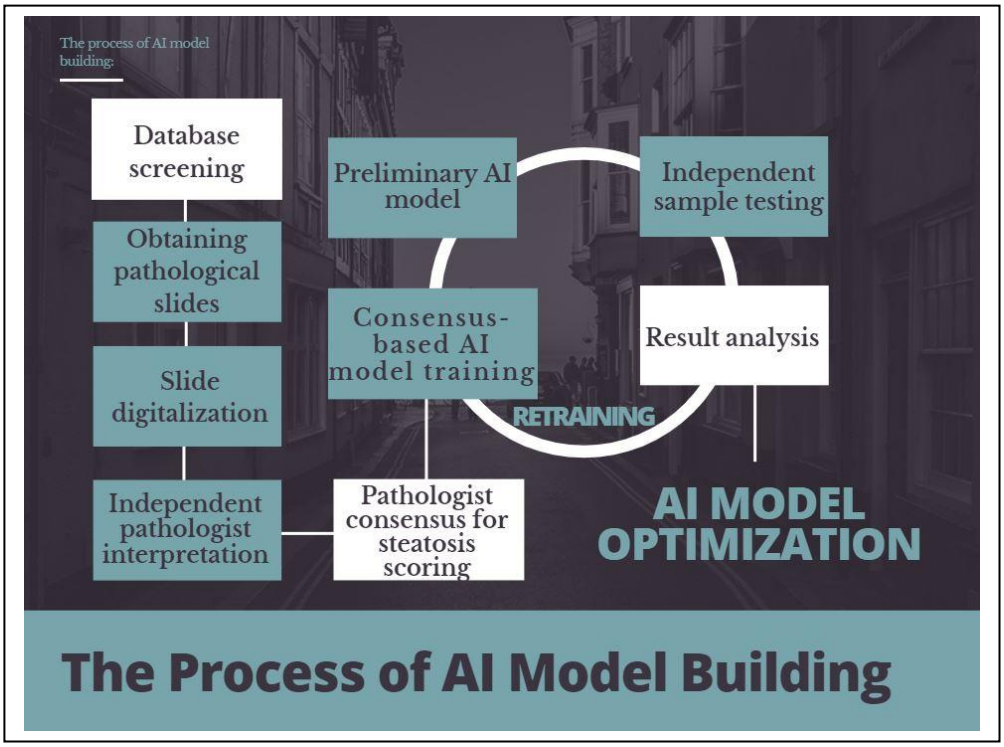


Figure 1. The process of AI model building

Obtaining pathological slides

After selecting sufficient liver pathology specimens from the database, we analyzed the selected pathology slides for quality control. Any malignancy in the liver tissues would cause a specimen to be excluded. In order to obtain high-quality pathological slides, we would perform a tissue section for the residual liver specimens (Figure 2). We initially obtained 100 pathology slides for the AI model training. If more slides were to be essential for retraining the primitive training model (Step 8 in Figure 1), we would gather an additional 100 slides.

Figure 2. Obtaining adequate pathological slides

Slide digitalization

We placed the liver sections onto the electronic image scanner, digitized them and archived the panoramic scanned image (Figure 3).

Independent pathologist interpretation

Two pathologists from Taichung Veterans General Hospital evaluated the severity level of hepatic steatosis in the liver tissue slides, and classified the severity levels into four groups: Class 0 (<5%), Class 1 (5-33%), Class 2 (>33-66%), and Class 3 (>66%).

Pathologist consensus for steatosis scoring

When the two pathologists' judgment on a slide differed, a consensus would be reached after a full discussion. The consensus score was treated as the standard score for the AI model training.

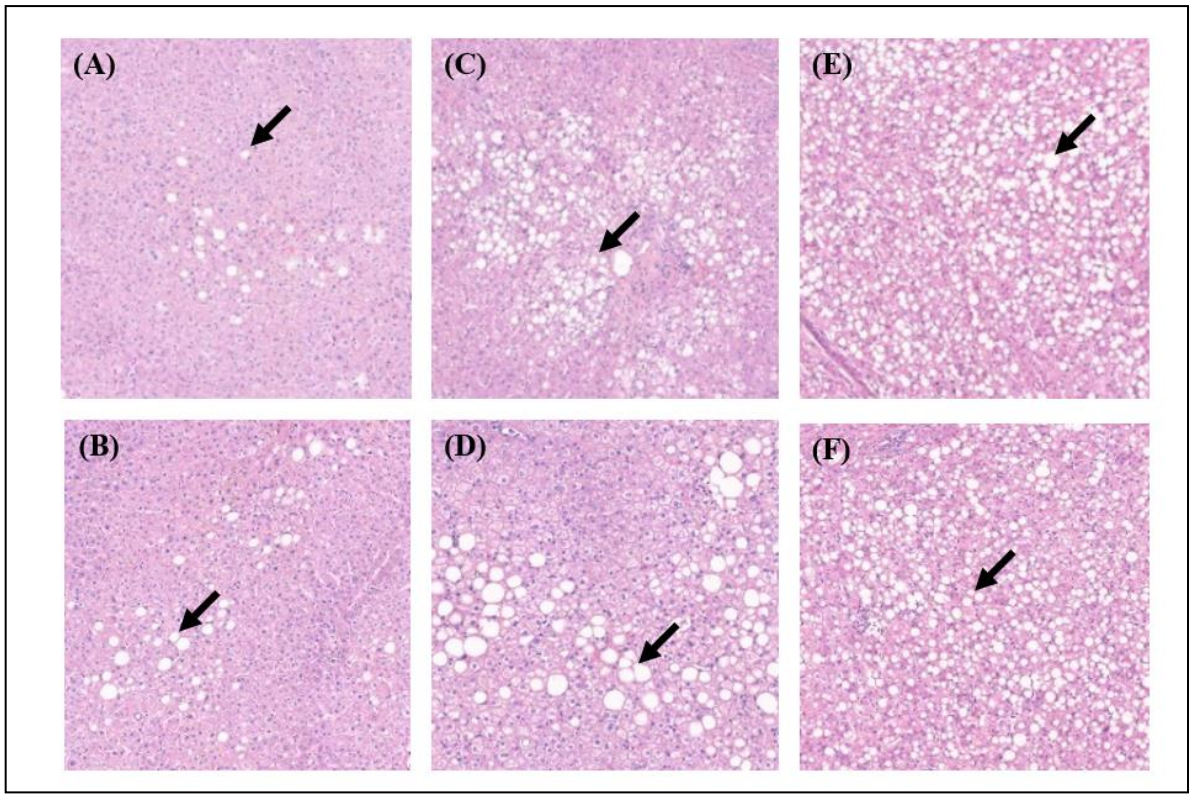


Figure 3. Arrows show the whitish round lesions of hepatic steatosis in pathological slides (100 times magnified): (A)(B) Class 1, (C)(D) Class 2, (E)(F) Class 3

Consensus-based AI model training

With the consensus scores evaluated by the pathologists now used as the standard answer to the training data, we trained a machine learning model with the Teachable Machine, which was developed by Google (https://teachablemachine.withgoogle.com). Teachable Machine utilizes JS MobileNet as its machine learning architecture, which is a lightweight deep convolutional neural network that provides an efficient model for mobile and embedded vision applications. Please refer to Supplementary File 1. for the source code of this machine learning model in Teachable Machine.

Preliminary AI model

After training by the consensus-based training data was completed, a primitive AI model was built for further independent sample testing.

Independent sample testing

After the initial model training was completed, we ran the independent test data through the model to obtain its predictions. At the same time, we respectively obtained both individual and consensus answers from the two pathologists.

Result analysis

We compared the divergence between the AI predictions and the consensus answers produced by the two pathologists.

AI model optimization

Based upon the differences between the predictions and answers, we reviewed and analyzed the effectiveness of the AI model, and continued improving the model according to the AI model's performance.

Statistical analysis

We used the confusion matrix for data analysis, as it is used for evaluating the performance of a classification model. The definitions surrounding statistical data analysis are as follows: True Positive (TP), a sample where the model correctly classified the positive class; False Positive (FP), a sample where the model incorrectly classified the positive class; False Negative (FN), a sample where the model incorrectly classified the negative class; True Negative (TN), a sample where the model correctly classified the negative class. Precision=TP/(TP+FP), the ratio of correct positive predictions to the total predicted positives. Recall=TP/ (TP+FN), the ratio of correct positive predictions to the total positive examples. F1-score=2*Precision *Recall/(Recall +Precision), F1-score is the weighted average of Precision and Recall.

Results and Discussions

Evaluation of each fatty liver class in the training model

As shown in Figure 4, after being trained with a total of 200 images from liver pathology slides, the model's accuracy growth curve (accuracy per epoch) eventually reached a stable performance of 0.8, although there was still room for further improvement. In addition, although the loss value was not high, the learning curve (loss per epoch) was not considered ideal. The accuracy of Class 1 was 0.93, which was the best of the four. Second in accuracy was Class 3 at 0.83, with Class 0 at 0.75, third. In this stage, the data necessary for evaluation provided by the Teachable Machine was not obtained from an independent test set; therefore, while it could be a valuable reference, it is not an absolute evaluation criterion. Due to the limited time taken to prepare the training data, the number of images in the training set was not a very large quantity, therefore the amount of image data in all classes should /needs to be increased in the future, particularly in the classes with a lower accuracy rate (e.g., Class 2). During the model training process, we found that an increase in the number of images in the training set could effectively improve the accuracy parameters. This will be one of the essential keys to improving this AI model in the future.

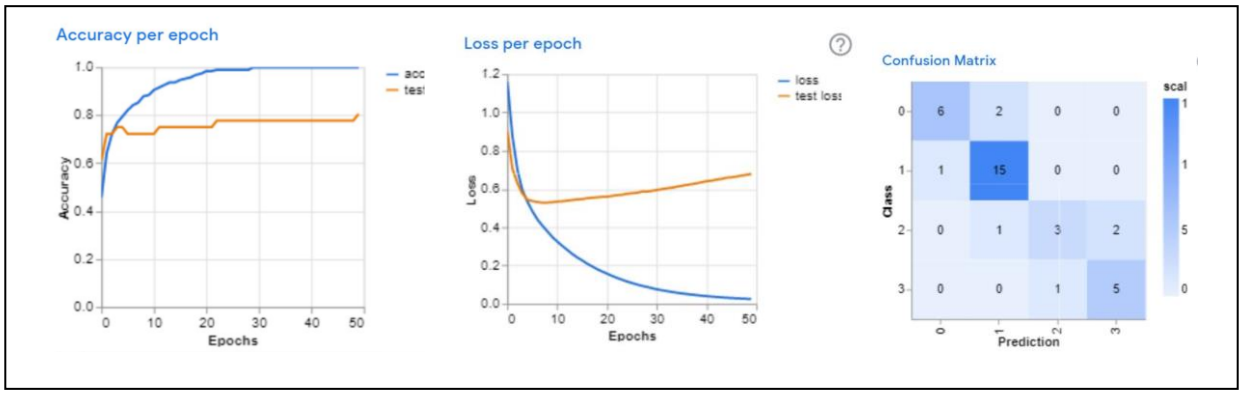
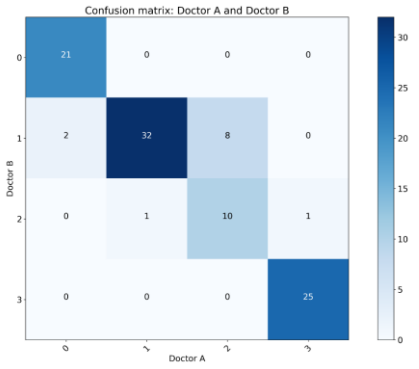


Figure 4. Performance of the training model

Scoring differences between the two pathologists

We took an additional 100 images from pathology slides to be used as the independent set,

and for the two pathologists to interpret individually. The results revealed that there were 12 different interpretations amongst the 100 images between the two pathologists (1 from Class 0, 8 from Class 1, 2 from Class 2, and 1 from Class 3). All of them differed by one level (e.g., Class 1 and Class 2), while there was no (0%) difference of two levels or more (e.g., Class 0 and Class 3). The 12 images with different diagnoses were then discussed by the two pathologists in order to reach a consensus, which were considered to be the standard answers for scoring. Pathologist A altered 5 interpretations of the former diagnosis, while pathologist B altered 7 interpretations of the former diagnosis, which does not indicate a significant difference. In the analysis of confusion matrix (Figure 5), although the weighted average of precision reached 0.91 between pathologists A and B, the precision in Class 2 turned out to be the worst (0.56). This shows that Class 2 is the most difficult to judge, possibly due to the fact that it lies directly in the middle of the levels of severity, which can be easily confused with Class 1 or 3. Additionally, the differences between the two pathologists can also influence the training and performance of our AI model.



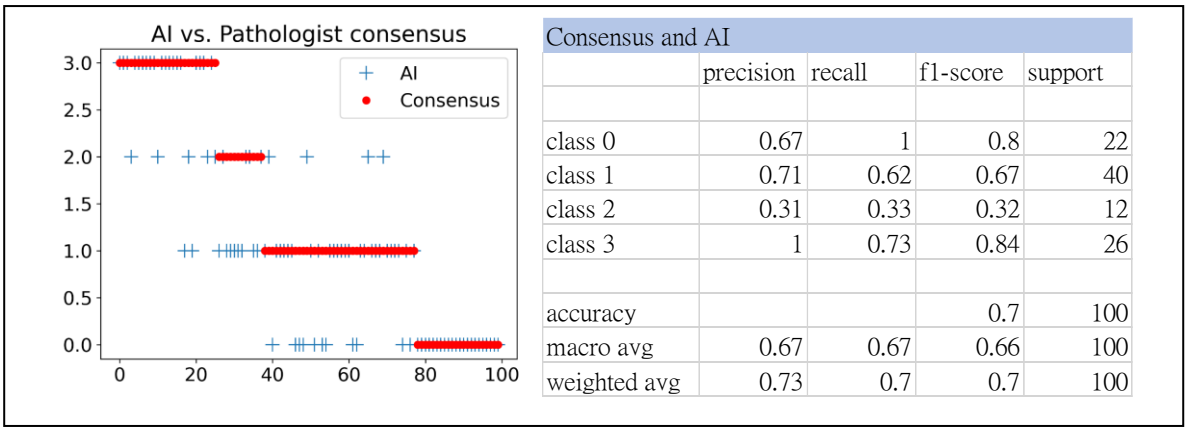
Doctor A and Doctor B				
	precision	recall	f1-score	support
class 0	0.91	1	0.95	21
class 1	0.97	0.76	0.85	42
class 2	0.56	0.83	0.67	12
class 3	0.96	1	0.98	25
accuracy			0.88	100
macro avg	0.85	0.9	0.86	100
weighted avg	0.91	0.88	0.88	100

Figure 5. Scoring differences between the two pathologists

Scoring differences between the AI model and the pathologist consensus

We tested the model using the independent test set that had been interpreted by the two pathologists, and compared its predictions with the consensus answers that the pathologists gave. The results showed that there were 29 different diagnoses within all 100 images, and most of them were off by only one level. Of the 29 images with incorrect predictions, 11 were in Class 0, 10 in

Class 1, 8 in Class 2, and none in Class 3. There were 2 images which were wrong by two levels and none wrong by three levels. The two which were incorrect by two levels were both Class 3, but misclassified into Class 1. Moreover, in 26 of the 100 images from the independent test set with the standard answers of Class 3, seven of the 26 were incorrectly interpreted by the AI model. Of the 7 images, 5 were misinterpreted as Class 2. The percentage of two-level misinterpretation was 7.7% (2/26), which is relatively low for a model with this amount of training data. However, if we consider a two-level-difference situation as being a severe misjudgment, it can indicate that there may be some unidentified bias between Classes 1 and 3. Moreover, in the data visualization graph (Figure 6), the precision of Class 2 is the worst, which reflects that the identification of Class 2 is in great need of both enhancement and improvement.



Consensus and AI				
	precision	recall	f1-score	support
class 0	0.67	1	0.8	22
class 1	0.71	0.62	0.67	40
class 2	0.31	0.33	0.32	12
class 3	1	0.73	0.84	26
accuracy			0.7	100
macro avg	0.67	0.67	0.66	100
weighted avg	0.73	0.7	0.7	100

Figure 6. Scoring differences between the pathologist consensus and the AI model

Cross analysis between the AI model, the consensus and each individual pathologist

We used the following four statistical models to perform cross analysis on the interpretation of the results. “Analysis Model 1” is the comparison between the consensus of the two pathologists and the interpretation of the AI model. "Analysis Model 2" is the comparison between Pathologist A and the AI model. "Analysis Model 3" is the comparison between Pathologist B and the AI model. "Analysis Model 4" is the comparison between Pathologist A and Pathologist B. The following table shows the results of the four analysis models in Recall, Precision, and F1-score.

Table 1. Recall in each analysis model.

Recall	Analysis Model 1	Analysis Model 2	Analysis Model 3	Analysis Model 4
	Consensus vs. AI	Dr. A vs. AI	Dr. B vs. AI	Dr. A vs. Dr. B
Class 0	1	1	1	1
Class 1	0.62	0.62	0.61	0.76
Class 2	0.33	0.33	0.28	0.83
Class 3	0.73	0.72	0.73	1

Table 2. Precision in each analysis model.

Precision	Analysis Model 1	Analysis Model 2	Analysis Model 3	Analysis Model 4
	Consensus vs. AI	Dr. A vs. AI	Dr. B vs. AI	Dr. A vs. Dr. B
Class 0	0.67	0.64	0.7	0.91
Class 1	0.71	0.74	0.57	0.97
Class 2	0.31	0.31	0.38	0.56
Class 3	1	0.95	1	0.96

Table 3. Fl-score in each analysis model.

Fl-score	Analysis Model 1	Analysis Model 2	Analysis Model 3	Analysis Model 4
	Consensus vs. AI	Dr. A vs. AI	Dr. B vs. AI	Dr. A vs. Dr. B
Class 0	0.8	0.78	0.82	0.95
Class 1	0.67	0.68	0.59	0.85
Class 2	0.32	0.32	0.32	0.67
Class 3	0.84	0.82	0.84	0.98

The standard answer to our current AI model training was based upon the consensus of the two pathologists. However, we discovered that different pathologist's judgments could differ, particularly when it came to moderate fatty liver (Class 2). This is understandable being that it's obviously easier to distinguish the differences between more extreme cases, and oppositely for the fuzzier middle ground (Classes 1 and 2), particularly when the severity level is close to the border line of the two classes (33% or 66%). If the standard answer is based upon human judgment, which can be rather uncertain, it can be expected that the model will be incapable of being 100% correct. Therefore, in addition to increasing the number of samples to avoid unidentified bias, there are two possible directions for model improvement: First, rather than striving to achieve 100% accuracy, we can change the goal of improvement to be the elimination of serious misinterpretation (incorrect by two levels or more), and consider one-level differences as a reasonable error, much like when it comes to cytological judgment regarding cervical cancer cells in current clinical practices. Second, we can reconsider whether the traditional classification is reasonable, by for example, changing the definition of moderate fatty liver, such as Class 2, in order to reduce the differences in the judgments of the pathologists. As shown in Figure 7, if we classify the degree of hepatic steatosis into only two levels, i.e. Class 0-1 & Class 2-3, the consistence and accuracy of this AI model can be generally improved.

Doctor A and Doctor B

	precision	recall	f1-score	support
Clasa 0-1	0.98	0.87	0.92	63
Clasa 2-3	0.82	0.97	0.89	37
accuracy			0.91	100
macro avg	0.9	0.92	0.91	100
weighted avg	0.92	0.91	0.91	100

Consensus and AI

	precision	recall	f1-score	support
Clasa 0-1	0.85	0.94	0.89	62
Clasa 2-3	0.8	0.74	0.8	38
accuracy			0.86	100
macro avg	0.86	0.84	0.85	100
weighted avg	0.86	0.86	0.86	100

Figure 7. The degree of hepatic steatosis was classified into two levels, i.e. Class 0-1 & Class 2-3.

Conclusions and Applications

Fatty liver has been a common disease in modern countries worldwide for many years. However, the application of AI on the diagnosis of fatty liver disease remains limited. Although previous AI studies were able to digitize and calculate the required data, the resulting technically "correct" information has shown a considerable difference with that of pathologists. For example, in the previous AI model (Forlano, 2020), the median percentage of hepatic steatosis in Class 3 was only 28.4%; however, the definition of Class 3 hepatic steatosis should be > 66%. In addition to the above-mentioned discrepancy between the exact percentage and the clinical definition in fatty liver classification, we have found that some tricky problems exist before applying AI to the diagnosis of fatty liver. For example, the semi-quantitative definitions of scoring classes in fatty liver diagnosis. In the previous definition of moderate fatty liver (Class 2), 33-66% of hepatic steatosis accumulation should be estimated in the slide, but what is actually the percentage within the range of 33-66% may seem more or less to the human eye. Furthermore, scoring differences between the results given by different pathologists are not so rare. These delicate problems should be considered the important study gaps which will require great effort and hard work in order to resolve. Nonetheless, it seems that there is an unknown "common standard" when it comes to humans estimating the severity of fatty liver. Different pathologists' interpretations may differ slightly, but never on a huge scale, e.g., no interpretation of liver slides from our two pathologists differed by more than one level. With that in mind, the model we built was based upon the consensus diagnosis provided by pathologists, and therefore should be able to produce interpretations closer to those of pathologists in the real world. Although our model still has a long way to go before it can be used in clinical practice, we still achieved a goal of having only 2% of the results varying above one levels, and none more than two levels. It is apparent that if there is an investment of more time, resources and technology, as well as in-depth discussions with pathologists to optimize our model, using AI to diagnose fatty liver is definitely feasible, and

worth the effort to continue its research. Through this experimental study, we are grateful to have learned from these practical experiences, which are nowhere to be found in previously given lectures or available literature. We now feel that we have a clearer understanding of the interpretation of both the human brain and AI.

References

Chalasani, N., Z. Younossi, J. E. Lavine, M. Charlton, K. Cusi, M. Rinella, S. A. Harrison, E. M. Brunt and A. J. Sanyal (2018). "The diagnosis and management of nonalcoholic fatty liver disease: Practice guidance from the American Association for the Study of Liver Diseases." Hepatology 67(1): 328-357.

Chalasani, N., Z. Younossi, J. E. Lavine, A. M. Diehl, E. M. Brunt, K. Cusi, M. Charlton and A. J. Sanyal (2012). "The diagnosis and management of non-alcoholic fatty liver disease: practice Guideline by the American Association for the Study of Liver Diseases, American College of Gastroenterology, and the American Gastroenterological Association." Hepatology 55(6): 2005-2023.

Chen, C. H., M. H. Huang, J. C. Yang, C. K. Nien, C. C. Yang, Y. H. Yeh and S. K. Yueh (2007). "Prevalence and etiology of elevated serum alanine aminotransferase level in an adult population in Taiwan." J Gastroenterol Hepatol 22(9): 1482-1489.

Cusi, K. (2012). "Role of obesity and lipotoxicity in the development of nonalcoholic steatohepatitis: pathophysiology and clinical implications." Gastroenterology 142(4): 711-725 e716.

Forlano, R., B. H. Mullish, N. Giannakeas, J. B. Maurice, N. Angkathunyakul, J. Lloyd, A. T. Tzallas, M. Tsipouras, M. Yee, M. R. Thursz, R. D. Goldin and P. Manousou (2020). "High-Throughput, Machine Learning-Based Quantification of Steatosis, Inflammation, Ballooning, and Fibrosis in Biopsies From Patients With Nonalcoholic Fatty Liver Disease." Clin

Gastroenterol Hepatol 18(9): 2081-2090 e2089.

Kleiner, D. E., E. M. Brunt, M. Van Natta, C. Behling, M. J. Contos, O. W. Cummings, L. D. Ferrell, Y. C. Liu, M. S. Torbenson, A. Unalp-Arida, M. Yeh, A. J. McCullough, A. J. Sanyal and N. Nonalcoholic Steatohepatitis Clinical Research (2005). "Design and validation of a histological scoring system for nonalcoholic fatty liver disease." Hepatology 41(6): 1313-1321.

Le Berre, C., W. J. Sandborn, S. Aridhi, M. D. Devignes, L. Fournier, M. Smail-Tabbone, S. Danese and L. Peyrin-Biroulet (2020). "Application of Artificial Intelligence to Gastroenterology and Hepatology." Gastroenterology 158(1): 76-94 e72.

Lee, T. Y., J. C. Wu, S. H. Yu, J. T. Lin, M. S. Wu and C. Y. Wu (2017). "The occurrence of hepatocellular carcinoma in different risk stratifications of clinically noncirrhotic nonalcoholic fatty liver disease." Int J Cancer 141(7): 1307-1314.

Li, J., B. Zou, Y. H. Yeo, Y. Feng, X. Xie, D. H. Lee, H. Fujii, Y. Wu, L. Y. Kam, F. Ji, X. Li, N. Chien, M. Wei, E. Ogawa, C. Zhao, X. Wu, C. D. Stave, L. Henry, S. Barnett, H. Takahashi, N. Furusyo, Y. Eguchi, Y. C. Hsu, T. Y. Lee, W. Ren, C. Qin, D. W. Jun, H. Toyoda, V. W. Wong, R. Cheung, Q. Zhu and M. H. Nguyen (2019). "Prevalence, incidence, and outcome of non-alcoholic fatty liver disease in Asia, 1999-2019: a systematic review and meta-analysis." Lancet Gastroenterol Hepatol 4(5): 389-398.

Satapathy, S. K. and A. J. Sanyal (2015). "Epidemiology and Natural History of Nonalcoholic Fatty Liver Disease." Semin Liver Dis 35(3): 221-235.

Sheka, A. C., O. Adeyi, J. Thompson, B. Hameed, P. A. Crawford and S. Ikramuddin (2020). "Nonalcoholic Steatohepatitis: A Review." JAMA 323(12): 1175-1183.

Younossi, Z. M., M. Otgonsuren, L. Henry, C. Venkatesan, A. Mishra, M. Erario and S. Hunt (2015). "Association of nonalcoholic fatty liver disease (NAFLD) with hepatocellular carcinoma (HCC) in the United States from 2004 to 2009." Hepatology 62(6): 1723-1730.

Supplementary File 1. The source code of this machine learning model in Teachable Machine (https://github.com/googlecreativelab/teachablemachine-community/blob/master/libraries/image/src/teachable-mobilenet.ts)

```
/**
       * @license
       * Copyright 2019 Google LLC. All Rights Reserved.
       * Licensed under the Apache License, Version 2.0 (the "License");
       * you may not use this file except in compliance with the License.
       * You may obtain a copy of the License at
       *
       * http://www.apache.org/licenses/LICENSE-2.0
       *
       * Unless required by applicable law or agreed to in writing, software
       * distributed under the License is distributed on an "AS IS" BASIS,
       * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
       * See the License for the specific language governing permissions and
       * limitations under the License.
       * =============================================================================
       */
      import * as tf from '@tensorflow/tfjs';
      import { util, Rank } from '@tensorflow/tfjs';
      import { capture } from './utils/tf';
      import { TensorContainer } from '@tensorflow/tfjs-core/dist/tensor_types';
      import { CustomCallbackArgs } from '@tensorflow/tfjs';
      import { CustomMobileNet,
          Metadata,
          loadTruncatedMobileNet,
          ClassifierInputSource,
          ModelOptions
      } from './custom-mobilenet';
      import * as seedrandom from 'seedrandom';
      import { Initializer } from '@tensorflow/tfjs-layers/dist/initializers';
      const VALIDATION_FRACTION = 0.15;
      export interface TrainingParameters {
          denseUnits: number;
          epochs: number;
          learningRate: number;
          batchSize: number;
      }
      interface Sample {
          data: Float32Array;
          label: number[];
      }
      // tslint:disable-next-line:no-any
      const isTensor = (c: any): c is tf.Tensor =>
          typeof c.dataId === 'object' && typeof c.shape === 'object';
      /**
       * Converts an integer into its one-hot representation and returns
       * the data as a JS Array.
       */
      function flatOneHot(label: number, numClasses: number) {
          const labelOneHot = new Array(numClasses).fill(0) as number[];
          labelOneHot[label] = 1;
          return labelOneHot;
      }
      /**
       * Shuffle an array of Float32Array or Samples using Fisher-Yates algorithm
       * Takes an optional seed value to make shuffling predictable
       */
      function fisherYates(array: Float32Array[] | Sample[], seed?: seedrandom.prng) {
          const length = array.length;
          // need to clone array or we'd be editing original as we goo
          const shuffled = array.slice();
          for (let i = (length - 1); i > 0; i -= 1) {
              let randomIndex ;
              if (seed) {
                  randomIndex = Math.floor(seed() * (i + 1));
              }
              else {
```

```
            randomIndex = Math.floor(Math.random() * (i + 1));
        }
        [shuffled[i], shuffled[randomIndex]] = [shuffled[randomIndex],shuffled[i]];
    }
    return shuffled;
}
export class TeachableMobileNet extends CustomMobileNet {
    /**
     * the training model for transfer learning
     */
    protected trainingModel: tf.LayersModel;
    /**
     * Training and validation datasets
     */
    private trainDataset: tf.data.Dataset<TensorContainer>;
    private validationDataset: tf.data.Dataset<TensorContainer>;
    private __stopTrainingResolve: () => void;
    // private __stopTrainingReject: (error: Error) => void;
    // Number of total samples
    private totalSamples = 0;
    // Array of all the examples collected
    public examples: Float32Array[][] = [];
    // Optional seed to make shuffling of data predictable
    private seed: seedrandom.prng;
    constructor(truncated: tf.LayersModel, metadata: Partial<Metadata>) {
        super(tf.sequential(), metadata);
        // the provided model is the truncated mobilenet
        this.truncatedModel = truncated;
    }
    public get asSequentialModel() {
        return this.model as tf.Sequential;
    }
    /**
     * has the teachable model been trained?
     */
    public get isTrained() {
        return !!this.model && this.model.layers && this.model.layers.length > 2;
    }
    /**
     * has the dataset been prepared with all labels and samples processed?
     */
    public get isPrepared() {
        return !!this.trainDataset;
    }
    /**
     * how many classes are in the dataset?
     */
    public get numClasses(): number {
        return this._metadata.labels.length;
    }
    /**
     * Add a sample of data under the provided className
     * @param className the classification this example belongs to
     * @param sample the image / tensor that belongs in this classification
     */
    // public async addExample(className: number, sample: HTMLCanvasElement | tf.Tensor) {
    public async addExample(className: number, sample: HTMLImageElement | HTMLCanvasElement | tf.Tensor)
{
        const cap = isTensor(sample) ? sample : capture(sample);
        const example = this.truncatedModel.predict(cap) as tf.Tensor;
        const activation = example.dataSync() as Float32Array;
        cap.dispose();
        example.dispose();
        // save samples of each class separately
        this.examples[className].push(activation);
        // increase our sample counter
        this.totalSamples++;
    }
    /**
     * Classify an input image / Tensor with your trained model. Return all results.
     * @param image the input image / Tensor to classify against your model
     * @param topK how many of the top results do you want? defautls to 3
     */
    public async predict(image: ClassifierInputSource, flipped = false) {
```

```
    if (!this.model) {
        throw new Error('Model has not been trained yet, called train() first');
    }
    return super.predict(image, flipped);
}
/**
 * Classify an input image / Tensor with your trained model. Return topK results
 * @param image the input image / Tensor to classify against your model
 * @param maxPredictions how many of the top results do you want? defautls to 3
 * @param flipped whether to flip an image
 */
public async predictTopK(image: ClassifierInputSource, maxPredictions = 10, flipped = false, ) {
    if (!this.model) {
        throw new Error('Model has not been trained yet, called train() first');
    }
    return super.predictTopK(image, maxPredictions, flipped);
}
/**
 * process the current examples provided to calculate labels and format
 * into proper tf.data.Dataset
 */
public prepare() {
    for (const classes in this.examples){
        if (classes.length === 0) {
            throw new Error('Add some examples before training');
        }
    }
    const datasets = this.convertToTfDataset();
    this.trainDataset = datasets.trainDataset;
    this.validationDataset = datasets.validationDataset;
}
/**
 * Process the examples by first shuffling randomly per class, then adding
 * one-hot labels, then splitting into training/validation datsets, and finally
 * sorting one last time
 */
private convertToTfDataset() {
    // first shuffle each class individually
    // TODO: we could basically replicate this by insterting randomly
    for (let i = 0; i < this.examples.length; i++) {
        this.examples[i] = fisherYates(this.examples[i], this.seed) as Float32Array[];
    }
    // then break into validation and test datasets
    let trainDataset: Sample[] = [];
    let validationDataset: Sample[] = [];
    // for each class, add samples to train and validation dataset
    for (let i = 0; i < this.examples.length; i++) {
        const y = flatOneHot(i, this.numClasses);
        const classLength = this.examples[i].length;
        const numValidation = Math.ceil(VALIDATION_FRACTION * classLength);
        const numTrain = classLength - numValidation;
        const classTrain = this.examples[i].slice(0, numTrain).map((dataArray) => {
            return { data: dataArray, label: y };
        });
        const classValidation = this.examples[i].slice(numTrain).map((dataArray) => {
            return { data: dataArray, label: y };
        });
        trainDataset = trainDataset.concat(classTrain);
        validationDataset = validationDataset.concat(classValidation);
    }
    // finally shuffle both train and validation datasets
    trainDataset = fisherYates(trainDataset, this.seed) as Sample[];
    validationDataset = fisherYates(validationDataset, this.seed) as Sample[];
    const trainX = tf.data.array(trainDataset.map(sample => sample.data));
    const validationX = tf.data.array(validationDataset.map(sample => sample.data));
    const trainY = tf.data.array(trainDataset.map(sample => sample.label));
    const validationY = tf.data.array(validationDataset.map(sample => sample.label));
    // return tf.data dataset objects
    return {
        trainDataset: tf.data.zip({ xs: trainX,  ys: trainY}),
        validationDataset: tf.data.zip({ xs: validationX,  ys: validationY})
    };
}
/**
```

```
     * Saving `model`'s topology and weights as two files
     * (`my-model-1.json` and `my-model-1.weights.bin`) as well as
     * a `metadata.json` file containing metadata such as text labels to be
     * downloaded from browser.
     * @param handlerOrURL An instance of `IOHandler` or a URL-like,
     * scheme-based string shortcut for `IOHandler`.
     * @param config Options for saving the model.
     * @returns A `Promise` of `SaveResult`, which summarizes the result of
     * the saving, such as byte sizes of the saved artifacts for the model's
     *   topology and weight values.
     */
    public async save(handlerOrURL: tf.io.IOHandler | string, config?: tf.io.SaveConfig):
Promise<tf.io.SaveResult> {
        return this.model.save(handlerOrURL, config);
    }
    /**
     * Train your data into a new model and join it with mobilenet
     * @param params the parameters for the model / training
     * @param callbacks provide callbacks to receive training events
     */
    public async train(params: TrainingParameters, callbacks: CustomCallbackArgs = {}) {
        // Add callback for onTrainEnd in case of early stop
        const originalOnTrainEnd = callbacks.onTrainEnd || (() => {});
        callbacks.onTrainEnd = (logs: tf.Logs) => {
            if (this.__stopTrainingResolve) {
                this.__stopTrainingResolve();
                this.__stopTrainingResolve = null;
            }
            originalOnTrainEnd(logs);
        };

        // Rest of trian function
        if (!this.isPrepared) {
            this.prepare();
        }
        const numLabels = this.getLabels().length;
        util.assert(
            numLabels === this.numClasses,
            () => `Can not train, has ${numLabels} labels and ${this.numClasses} classes`);
        const inputShape = this.truncatedModel.outputs[0].shape.slice(1); // [ 7 x 7 x 1280]
        const inputSize = tf.util.sizeFromShape(inputShape);
        // in case we need to use a seed for predictable training
        let varianceScaling: Initializer;
        if (this.seed) {
            varianceScaling = tf.initializers.varianceScaling({ seed: 3.14});
        }
        else {
            varianceScaling = tf.initializers.varianceScaling({});
        }
        this.trainingModel = tf.sequential({
            layers: [
                tf.layers.dense({
                    inputShape: [inputSize],
                    units: params.denseUnits,
                    activation: 'relu',
                    kernelInitializer: varianceScaling, // 'varianceScaling'
                    useBias: true
                }),
                tf.layers.dense({
                    kernelInitializer: varianceScaling, // 'varianceScaling'
                    useBias: false,
                    activation: 'softmax',
                    units: this.numClasses
                })
            ]
        });
        const optimizer = tf.train.adam(params.learningRate);
        // const optimizer = tf.train.rmsprop(params.learningRate);
        this.trainingModel.compile({
            optimizer,
            // loss: 'binaryCrossentropy',
            loss: 'categoricalCrossentropy',
            metrics: ['accuracy']
        });
```

```
    if (!(params.batchSize > 0)) {
        throw new Error(
        `Batch size is 0 or NaN. Please choose a non-zero fraction`
        );
    }
    const trainData = this.trainDataset.batch(params.batchSize);
    const validationData = this.validationDataset.batch(params.batchSize);
    // For debugging: check for shuffle or result from trainDataset
    /*
    await trainDataset.forEach((e: tf.Tensor[]) => {
        console.log(e);
    })
    */
    const history = await this.trainingModel.fitDataset(trainData, {
        epochs: params.epochs,
        validationData,
        callbacks
    });
    const jointModel = tf.sequential();
    jointModel.add(this.truncatedModel);
    jointModel.add(this.trainingModel);
    this.model = jointModel;
    optimizer.dispose(); // cleanup of memory
    return this.model;
}
/*
 * Setup the exampls array to hold samples per class
 */
public prepareDataset() {
    for (let i = 0; i < this.numClasses; i++) {
        this.examples[i] = [];
    }
}
public setLabel(index: number, label: string) {
    this._metadata.labels[index] = label;
}
public setLabels(labels: string[]) {
    this._metadata.labels = labels;
    this.prepareDataset();
}
public getLabel(index: number) {
    return this._metadata.labels[index];
}
public getLabels() {
    return this._metadata.labels;
}
public setName(name: string) {
    this._metadata.modelName = name;
}
public getName() {
    return this._metadata.modelName;
}
public stopTraining() {
    const promise = new Promise((resolve, reject) => {
        this.trainingModel.stopTraining = true;
        this.__stopTrainingResolve = resolve;
        // this.__stopTrainingReject = reject;
    });

    return promise;
}
public dispose() {
    this.trainingModel.dispose();
    super.dispose();
}
/*
 * Calculate each class accuracy using the validation dataset
 */
public async calculateAccuracyPerClass() {
    const validationXs = this.validationDataset.mapAsync(async (dataset: TensorContainer) => {
        return (dataset as { xs: TensorContainer, ys: TensorContainer}).xs;
    });
    const validationYs = this.validationDataset.mapAsync(async (dataset: TensorContainer) => {
        return (dataset as { xs: TensorContainer, ys: TensorContainer}).ys;
```

```
        });
        // we need to split our validation data into batches in case it is too large to fit in memory
        const batchSize = Math.min(validationYs.size, 32);
        const iterations = Math.ceil(validationYs.size / batchSize);
        const batchesX = validationXs.batch(batchSize);
        const batchesY = validationYs.batch(batchSize);
        const itX = await batchesX.iterator();
        const itY = await batchesY.iterator();
        const allX = [];
        const allY = [];
        for (let i = 0; i < iterations; i++) {
            // 1. get the prediction values in batches
            const batchedXTensor = await itX.next();
            const batchedXPredictionTensor = this.trainingModel.predict(batchedXTensor.value) as
tf.Tensor;
            const argMaxX = batchedXPredictionTensor.argMax(1); // Returns the indices of the max values
along an axis
            allX.push(argMaxX);
            // 2. get the ground truth label values in batches
            const batchedYTensor = await itY.next();
            const argMaxY = batchedYTensor.value.argMax(1); // Returns the indices of the max values along
an axis
            allY.push(argMaxY);
            // 3. dispose of all our tensors
            batchedXTensor.value.dispose();
            batchedXPredictionTensor.dispose();
            batchedYTensor.value.dispose();
        }
        // concatenate all the results of the batches
        const reference = tf.concat(allY); // this is the ground truth
        const predictions = tf.concat(allX); // this is the prediction our model is guessing
        // only if we concatenated more than one tensor for preference and reference
        if (iterations !== 1) {
            for (let i = 0; i < allX.length; i++) {
                allX[i].dispose();
                allY[i].dispose();
            }
        }
        return { reference, predictions };
    }
    /*
     * optional seed for predictable shuffling of dataset
     */
    public setSeed(seed: string) {
        this.seed = seedrandom(seed);
    }
}
export async function createTeachable(metadata: Partial<Metadata>, modelOptions?: ModelOptions) {
    const mobilenet = await loadTruncatedMobileNet(modelOptions);
    return new TeachableMobileNet(mobilenet, metadata);
}
```

【評語】090019

整體研究主題鮮明有趣，雖然實驗經費有限，以至於能使用的分析工具有限，但建議在拍照時，顏色呈現可以加入控制組，避免自動調節色彩以至於誤導實驗結果。對於實驗原理的闡述及結果的呈現，可再用更精確的文字多加描述、以及更直觀可理解的圖表呈現，可讓讀者更容易理解該實驗設計的結果。最後應用推廣部分，可更多的說明該實驗對於市場上商品有何特殊創新試驗或貢獻等，會更加完整。