Efficient Convolutional Neural Networks for Automated Cognitive Diagnosis

1st Connor Pearson
Computing Science Department
University of East Anglia
Norwich, England
connorpearson.cs@gmail.com

2rd Beatriz De La Iglesia
School of Computing Sciences
University of East Anglia
Norwich, England
B.Iglesia@uea.ac.uk

3rd Saber Sami
Norwich Medical School
University of East Anglia
Norwich, England
S.Sami@uea.ac.uk

***Abstract*—Digitization has transformed diagnostic methods in several healthcare sectors. The standard cognitive assessment tests, evaluate cognitive impairment including early stages that can potentially progress to Alzheimer's Disease. However, it poses challenges due to manual administration.**

Here we propose using a novel convolutional neural network described here as CogniNet and compare its performance with leading doodle recognition transfer learning models to automate the visuospatial aspect of cognitive tests. Based on our CogniNet model we developed a web-application on the Laravel framework with enhanced accessibility and security features.

Our convolutional neural network achieved 91.5% accuracy, while the EfficientNet And MobileNet transfer learning models reached 87.5% and 85.5% respectively.

I. Introduction

Dementia presents an increasingly significant challenge for healthcare systems globally. A 2015 study estimated that 650,000 individuals were diagnosed with dementia in the UK alone, a number expected to rise to 1,351,000 by 2040 [1]. With the annual cost of dementia care already impacting the NHS at £24.2 billion [2], it is important to explore strategies to mitigate these expenses as demand escalates.

Digitization of diagnostic tests, such as the Addenbrooke's Cognitive Examination (ACE-III), offers a promising avenue for cost reduction. The ACE-III is recognized for its high accuracy, ranging from 82% to 97% in detecting dementia and 75% to 77% for Mild Cognitive Impairment (MCI) alone [3] . It is especially noted for its sensitivity in early dementia detection. This paper focuses on transforming the visuospatial component for diagnostic tests into a digital format.

Our methodology involves developing a bespoke CNN alongside exploring efficient transfer learning models, including EfficientNet and MobileNet, which are noted for their favorable performance to parameter count ratio [4]. This makes these models ideal for web-applications intended for widespread use where computing resources are limited.

This study primarily utilizes CNN-based models due to their proven effectiveness in Optical Character Recognition (OCR) tasks enabling concurrent learning of feature extraction and classification layers [5]. This allows these models to produce highly accurate outputs based on extracted image features.

We have integrated these models with a web interface, enabling patients to undergo testing remotely in a setting that best fits their needs. This approach can facilitate easier access to regular testing as well as enhanced diagnostic accuracy and reduced costs associated with traditional methods.

II. Aims

This project aims to create an online solution that allows patients to take a test similar to that of the visuospatial portion of pen and paper diagnostic tests intending to provide insights into a person's cognitive health.

This is carried out using CNNs, both bespoke and pre-trained transfer learning based models. Our focus on the EfficientNet and MobileNet models is based on their demonstrated strong parameter count to performance ratio, as shown in a recent comparative analysis by Yuan Yang et al. [4]

With these aims in consideration a more granular set of objectives can be derived.

A. Objectives

1) Allow us to batch test a series of models with varying hyper-parameters.
   a) Scalable approach covering hyper-parameter testing of large range of model architectures: i) epochs ii) structures iii) kernel-sizes iv) dropout
   b) Capable of compensating for varying data structure requirements (seen in table I) without compromising image integrity.
   c) Allow for standard Command-Line Interface (CLI) approach to development with common features such as a standard 'help' print screen for new users, as well as also being platform agnostic.
2) Create a series of high-performance models capable of classifying doodles into 25 separate select categories from the Google Quick, Draw! dataset.
   a) Training to utilise a grid search style approach to trying varying model hyper-parameters ran in series over a span of a year, the aim being to find the most performant of each model sub-group (Bespoke CNN, EfficientNet and MobileNet).
   b) Training to make use of standard training set / testing set split with a hold-out set of roughly 30%.

c) Final evaluations of the model will be ran on a series of real-world production data taken from test sessions on the live site.

3) Produce a web-application to serve as a platform for our digital test using the Laravel 11 framework.
   a) Site uses Tailwind/CSS as well as Vue in combination to provide a reactive and modern foundation to serve tests upon.
   b) Site features two distinct user types for patient and doctor, with scoped access to specific pages for each role.
   c) Site is platform agnostic, with support for common browser / device variations such a desktop and mobile.
   d) site uses the highest accuracy model with the least computational overhead (in terms of parameter count) to process tests in jobs within Laravel Horizon.

III. Implementation

A. Datasets & Pre-processing

Before beginning training, we first identify an large dataset that enables comprehensive training. With these considerations, the Google Quick, Draw! dataset became our main contender.

With strong relevance amongst many other datasets such as the MNIST dataset, Google Quick, Draw! proves its relevancy once more after its use previously in our earlier work in which we used a standard novel neural network to achieve an implementation accuracy of 70% [6].

1) Requirements: Table I shows the constraints that are present for each of the models used. With this, an informed decision can be made in the methods of which we intend to pre-process our data to fit. The notation of X, Y and Z are used to denote values that do not have constraints.

Model	Minimum Input Size	Minimum Channels
Bespoke CNN	X px × Y px	Z × Channels
MobileNet	32 px × 32 px	3 × Channels
EfficientNet	X px × Y px	3 × Channels

TABLE I: Input size specifications for CNN models

With the Google Quick, Draw! dataset we are provided with a series of '.npy' files, a format used by the common Python library NumPy. The files are split by doodle category i.e. 'ambulance.npy', 'basket.npy', 'apple.npy' and are provided in a pickled format by default.

Each file then contains a 3D array containing a list of 2D images for the associated category. These image arrays are structured as 28 px × 28 px doodles with a single Grayscale colour channel.

Google also provides this data in other formats such as raw '.ndjson' files as well as binary files, NumPy files were chosen for this project as they work natively with Python. [7]

B. Preparing Google Quick, Draw!

The transformation of the Google Quick, Draw! dataset for model training was managed by the script `src/process_datasets.py`, integrated into the main interface script `run.py`. Here is a streamlined overview of the processing steps:

1) Data Path Specification and Loading: The script uses the `dataset_path` parameter to locate and sequentially load `.npy` files, each named according to its content category.
2) Data Splitting and Labeling: Data within these files is divided into training and testing sets based on the `split` parameter. Each image is then labeled according to predefined categories, creating labeled datasets.
3) Image Format Adjustment:
   - Greyscale to RGB Conversion: Single-channel grayscale images are batched using NumPy to simulate RGB channels for models requiring 3-channel input.
   - Resizing Images: Images are resized to 32 px × 32 px as and when required using the Python Imaging Library (PIL) to meet the input dimensions required by some models.

This streamlined process ensures that the dataset is compatible with TensorFlow and various neural network architectures, facilitating effective training without compromising image data integrity or model performance.

C. Machine Learning Model

Our approach to identifying high-performance models involved using an automated script for batch training, which systematically trains AI models in related sets. This iterative process explored various model structures, epochs, learning rates, dropout rates, and kernel sizes within each training session. The resulting models were then evaluated and analyzed for key metrics including accuracy, parameter count, processing speed, and generalization.

The optimized models were subsequently saved in .keras files, each named according to its specific structure, as depicted in Figure 1.

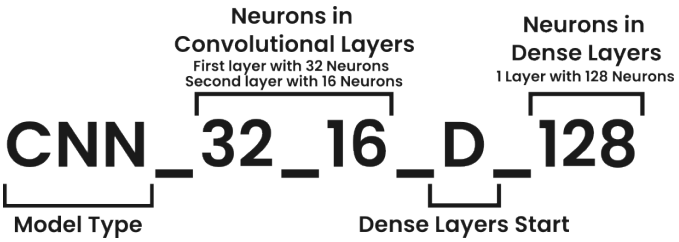


Fig. 1: Explanation of the model structure as shown in the filename of the stored model.

We conducted a grid search to explore all possible combinations of hyperparameters within a practical range tailored to our use case. Although hyperparameter optimization demands numerous evaluations of the convolutional neural network (CNN) and is thus computationally intensive, [8] the automation provided by our script significantly reduced the workload of this exhaustive search'.

D. Web-application

The development of the web-application took partly from the works of our previous paper and converted the previous framework from NodeJS to Laravel 11.

Additionally converting web pages previously using the PUG markup language alongside standard CSS to Vue files with Tailwind/CSS as the new CSS framework. Many site graphics were designed in Inkscape and exported as either PNG or SVG depending on the use case.

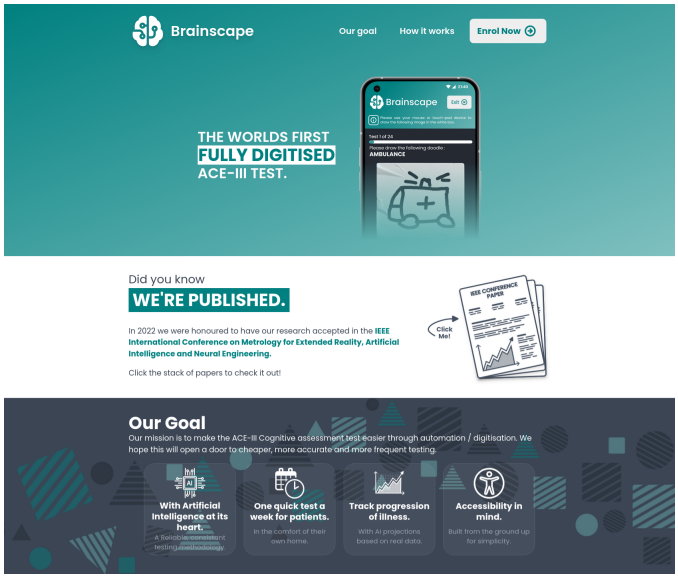


Fig. 2: Capture of re-designed home page for brainscape.uk.

Using Laravel Breeze as a foundation a standard Vue Laravel app with an Intertia front end implementation was generated. The web-application then had its web routes defined to allow for testing of the newly converted and added web pages. API routes where then defined for basic test functionality such as uploading doodles and managing user specific tests.

In addition a database was designed and implemented on the server-side to store user accounts as well as exam data and doodles alongside other metrics such as time-to-draw in milliseconds which is intended to be a metric that can be utilised later in the sites development to potentially better classify drawings.

The site was deployed via a DigitalOcean instance which could be managed via the Laravel Forge app. This allowed for the management of server deployments as well as additional server configuration such as nginx configuration and database connection management.

The server was provisioned with a simple dual core vCPU with 4GB of RAM. With such resources it is estimated to be capable of processing a steady flow of exam results using CogniNet in conjunction with Laravel Horizon which allows for the deployment of asynchronous jobs to a queue. In the event that site traffic increased or jobs prove to be too slow, the DigitalOcean service allows for easy and affordable vertical scaling of server resources.

E. Model Results

1) Model accuracies: A data split of 70/30 was used, meaning that 70% of the data would be used in training and the remaining 30% would be used in the evaluation of the models accuracy.

This provided the model with 2,415,230 doodles to train on and 1,035,097 doodles to test on. These sets of-course contain a mix of all 25 categories with shuffling of data added in to prevent any potential for order bias as well as improving generalisation. Transfer learning models were also pre-initialised with the imagenet weights to provide a stronger accuracy, CogniNet however used default weights.

In calculating the implementation accuracy of a model, a series of sessions were created via the web-application in which the doodles were converted into NumPy arrays which were then fed through the model. Testing on a total of 8 sessions from the web-application, calculating the percentage accuracy, summing the accuracy's of each session and then dividing the sum by the total number of session results.

In mathematical notation it would be a simple summation and division to calculate A like so :

$$A = \frac{1}{n}\sum_{i=1}^{n} s_i$$

where s_i represents the score from the i-th session, and n is the total number of sessions (in this case, 8).

Upon application of the scoring algorithm the script provided a selection of top performing models as shown in figure 3 where the implementation accuracy is shown.

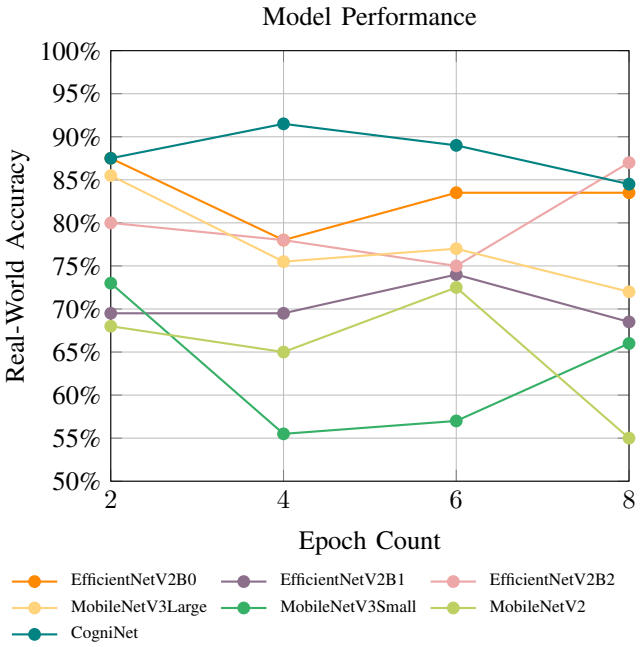


Fig. 3: Graph showing various real-world model accuracy's.

A more granular look at the final model scores can be seen in Table II where the two greatest scores are present

in bold. It is shown that the CogniNet model (the bespoke CNN) is the strongest model in the list, using the structure of 'CNN_32_16_D_128' with a 0.2 dropout layer. This model can be seen in both Figure 1 and also in 6.

Model	Epoch 2	Epoch 4	Epoch 6	Epoch 8
EfficientNetV2B0	87.5%	78.0%	83.5%	83.5%
EfficientNetV2B1	69.5%	69.5%	74.0%	68.5%
EfficientNetV2B2	80.0%	78.0%	75.0%	87.0%
MobileNetV3Large	85.5%	75.5%	77.0%	72.0%
MobileNetV3Small	73.0%	55.5%	57.0%	66.0%
MobileNetV2	68.0%	65.0%	72.5%	55.0%
CogniNet	87.5%	**91.5%**	**89.0%**	84.5%

TABLE II: Model performance at various epochs using our simple average scoring algorithm

Additionally, a 3D graph is provided below in figure 4 to aid in visualising the performance of models as we vary in model complexity and epoch count.

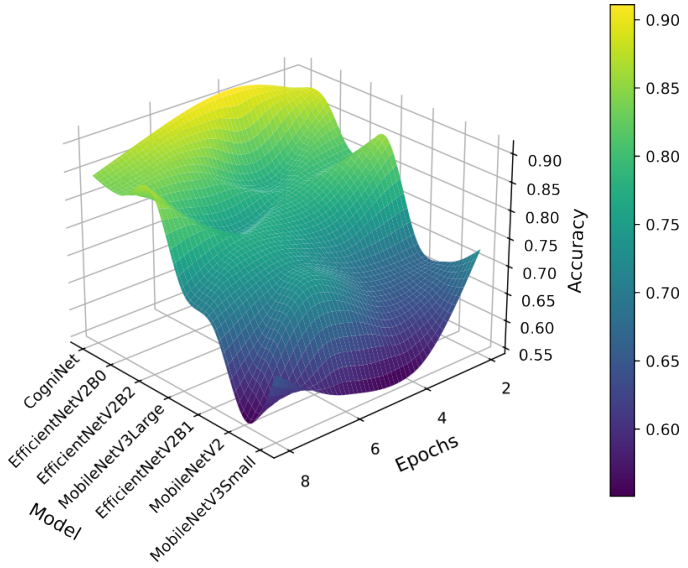


Fig. 4: Average scores using our custom scoring algorithm plotted as a 3d graph.

Finally, in addition to the use of the custom scoring algorithm a standard F1 score was taken for each of the 8 test sessions and an average F1 score was calculated for each model. These results are shown below in table III. The two accuracy's in bold represent the two highest F1 scores again showing CogniNet as the most performent.

2) Model Complexities: A separate metric by which to quantify our models is that of the models complexity. Often refereed to in context of the models parameter count and in this case, additionally its size.

Figure 5 shows first-hand the difference in complexity in models especially as we compare the CogniNet model against other pre-trained models.

The CogniNet model contains 327,485 parameters, while our most complex (and therefore computationally expensive) model, EfficientNetV2B2, comprises 30,548,871 parameters. To measure this complexity a simple formula was created :

Model Name	Epoch 2	Epoch 4	Epoch 6	Epoch 8
EfficientNetV2B0	84.167%	73.050%	79.433%	78.417%
EfficientNetV2B1	63.011%	62.200%	67.700%	62.910%
EfficientNetV2B2	75.833%	72.783%	68.650%	84.200%
MobileNetV3Large	81.533%	70.092%	70.783%	66.117%
MobileNetV3Small	66.200%	47.099%	49.254%	59.577%
MobileNetV2	62.018%	58.902%	66.183%	50.415%
CogniNet	83.749%	**89.000%**	**85.916%**	80.583%

TABLE III: Averaged F1 scores of various models at different batch sizes

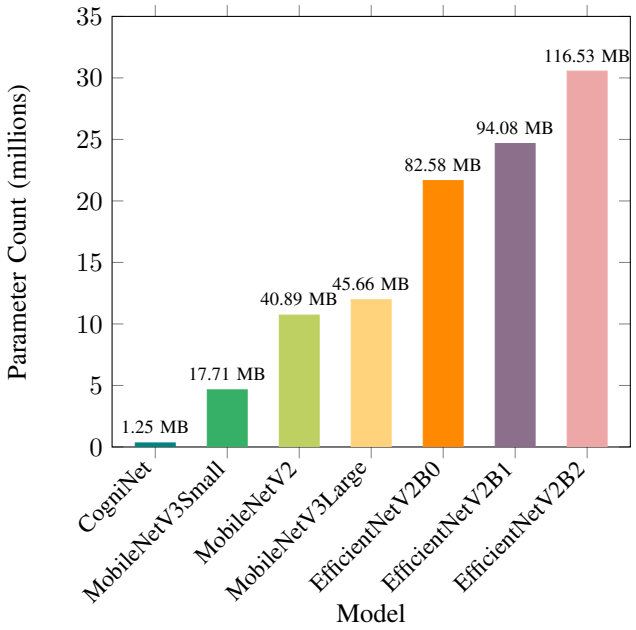


Fig. 5: Parameter count and size on disk in megabytes for various models

$$\text{Complexity Ratio} = \frac{\text{EfficientNetV2B2 Parameters}}{\text{CogniNet Parameters}} = \frac{30,548,871}{327,485} \approx 93.28$$

The complexity ratio calculation reveals that EfficientNetV2B2 is approximately 93.28 times more complex than CogniNet. Despite its higher complexity, EfficientNetV2B2 is outperformed by the simpler CogniNet model by 5.2%.

This additional model complexity also meant that while CogniNet would take on average 12ms to process an image, EfficientNetV2B2 would take an average of 20ms, making it 66.66% slower.

3) Model Architecture: In Figure 6 we see a graphical representation of our most performent CogniNet model. The graphic shows an image entering the network, then passing through the convolutional layers where feature extraction occurs. From there the image is flattened and passed into the dense layers where the model learns from these features.

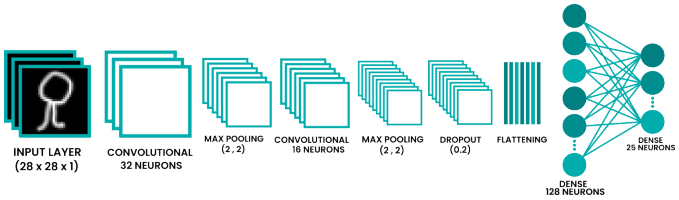
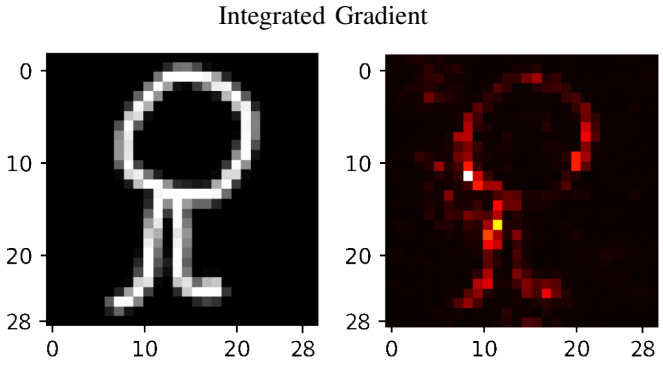


Fig. 6: Graphical representation of CogniNet models structure.

F. Visualised Learning

There are common methods of visualising the feature selection that occurs within CNNs. Using our script we deployed a couple of common methods to aid in understanding how these models process their inputs and ultimately determine a classification.

1) CogniNet Saliency: In the Figure 8 we see a sample of a saliency map from our CogniNet model. In this case a doodle of a tree. This image was passed through CogniNet using the tools made available by Google PAIR which allowed us to produce an Integrated Gradient map. This highlights in brighter colours which parts of the input contributed to its correct classification, showing what features the model uses from the doodle to classify it.



(a) Scaled (b) Saliency

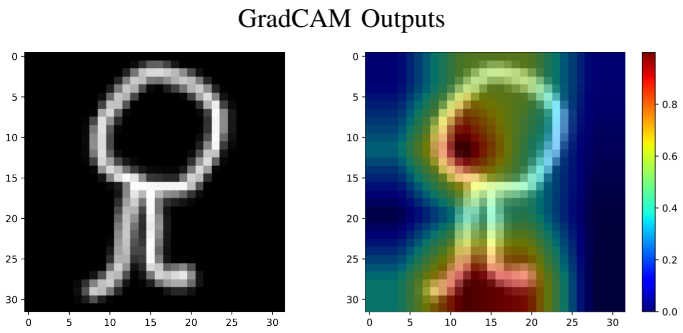
Fig. 7: Figure showing tree doodle input and computed integrated gradient map from CogniNet.

2) EfficientNetV2 GradCAM: With the architecture of the EfficientNetV2 we had access to some more advanced tools in the TensorFlow library which allowed for the creation of a GradCAM (Gradient-weighted Class Activation Mapping) output from the model.

The intention of the GradCAM output is to produce a heatmap of what parts of the input image influence activation's that ultimately determine the classification of the input. An example can be seen in Figure 8

3) Accuracy plotting: As mentioned previously, the script was provisioned to have a method of plotting accuracy's upon completion. This allowed us to determine which combination of hyper-parameters were working best.

An example of such as graph can be seen below in Figure 9. This example is from a small training session but outlines



(a) Input Image (b) Input & GradCAM overlayed

Fig. 8: Plot showing tree doodle input with GradCAM on EfficientNet model.

the degree in which these plots prove themselves informative in regards to the accuracy of a series of models.

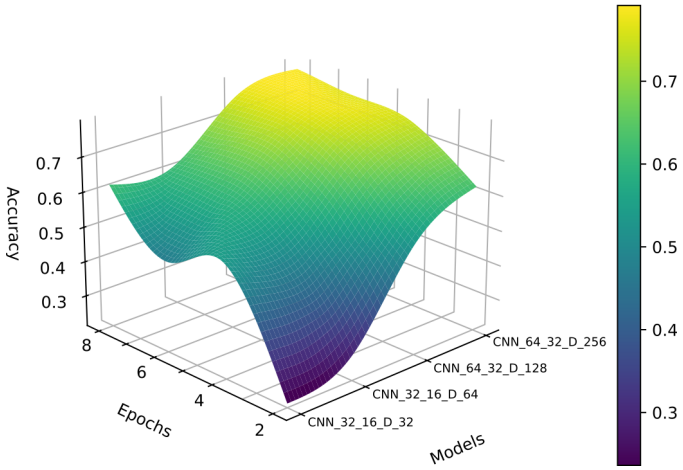


Fig. 9: Output graph showing theoretical accuracies of a training session.

For instance, in the case of this specific session the figure heavily implies that our most performent models will be in the 'CNN_64_32_D_256' subset particularly in the range of 8 to 10 epochs. With this data an informed decision on what models to train next and at what epoch range can be derived.

IV. Evaluation

A. Future Prospects

The prospects of the project can be defined in two parts, improved accuracy and increased breadth. Using stronger, more complex models a higher degree of accuracy could likely be achieved, but additionally combining these improved models with more relative data to the diagnosis of dementia could greatly increase our goal of detecting dementia.

1) Moving towards Long-Short Term Memory (LSTM) Models and Kolmogorov-Arnold Networks KANs: A natural next step for the project would be to move away from standard CNNs and towards the use of LSTM CNNs or alternatively

Convolutional Long Short-Term Memory Deep Neural Networks (CLDNN) models. These models are capable of learning from the temporal aspect of our drawing tasks. Allowing us to use the Google Quick, Draw! provided real-time drawing data to better predict doodle classes on a stroke-by-stroke basis.

These models have been shown to provide up to a '2-3% relative improvement in accuracy over the results of CNN and LSTM individual models.' [9] While this may seem as a minor difference however when your working with models reaching an average of 91% accuracy on real-world test data an additional 2 to 3% can make a significant contribution to a models viability.

In addition, recent research into Kolmogorov-Arnold Networks (KANs) have proven to provide roughly the equivalent performance to traditional CNNs which maintaining a significant reduction in model complexity as shown in a paper by Universidad de San Andrés, which states 'KANs seem to maintain accuracy with lower parameter count' which in this study showed a reduction in model complexity from '157k to 95k' parameters [10].

B. Complete coverage of the ACE-III

In addition to more advanced models, encapsulating the additional components of the ACE-III into the web applications digital examination will likely yield stronger accuracy's. The project covers the previously mentioned visuospatial component of the ACE-III, specifically a component of the test in which the patient is asked to copy a provided drawing as well as draw a clock without reference. Expanding upon the scope of our test to encapsulate further exam questions such as word recall would provide additional data points to train and interpret from.

V. Conclusions

This study underscores the potential of automating diagnostic testing through Artificial Intelligence (AI). It highlights the necessity of comprehensive model analysis when developing use-case-specific models, demonstrating that transfer learning models may not always be the most efficient or performant.

Quantifying the accuracy, and usability of a model in a real-world production environment requires more than just theoretical accuracy. Furthermore, this study successfully developed on our previous paper, achieving a 21.5% improvement on our previous model implementation accuracy of 70% using a novel neural network.

With further development and efficacy testing, CogniNet can be deployed in clinical trials to ensure it aligns with standard medical tests. Once strong efficacy is demonstrated, integration with existing healthcare systems can commence, starting with systems similar to the NHS Screening system.

References

[1] R. Wittenberg, B. Hu, C. Jagger, A. Kingston, M. Knapp, A. Comas-Herrera, D. King, A. Rehill, and S. Banerjee, "Projections of care for older people with dementia in England: 2015 to 2040," *Age and Ageing*, vol. 49, no. 2, pp. 264–269, 12 2019. [Online]. Available: https://doi.org/10.1093/ageing/afz154

[2] R. Wittenberg, M. Knapp, B. Hu, A. Comas-Herrera, D. King, A. Rehill, C. Shi, S. Banerjee, A. Patel, C. Jagger, and A. Kingston, "The costs of dementia in england," *International Journal of Geriatric Psychiatry*, vol. 34, no. 7, pp. 1095–1103, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/gps.5113

[3] L. Beishon, A. Batterham, T. Quinn, C. Nelson, R. Panerai, T. Robinson, and V. Haunton, "Addenbrooke's cognitive examination iii (ace-iii) and mini-ace for the detection of dementia and mild cognitive impairment," *Cochrane Database of Systematic Reviews*, 2019.

[4] Y. Yang, L. Zhang, M. Du, J. Bo, H. Liu, L. Ren, X. Li, and M. J. Deene, "A comparative analysis of eleven neural networks architectures for small datasets of lung images of covid-19 patients toward improved clinical decisions." *Comput Biol Med*, 2021.

[5] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[6] C. Pearson, B. De La Iglesia, and S. Sami, "Detecting cognitive decline using a novel doodle-based neural network," pp. 99–103, Oct 2022.

[7] Google Creative Lab, "Quick, draw! dataset," https://github.com/googlecreativelab/quickdraw-dataset, Access year, accessed: insert access date here.

[8] M. Wojciuk, Z. Swiderska-Chadaj, K. Siwek, and A. Gertych, "The role of hyperparameter optimization in fine-tuning of cnn models," *Available at SSRN 4087642*, 2022.

[9] A. Emam, M. Shalaby, M. A. Aboelazm, H. E. A. Bakr, and H. A. Mansour, "A comparative study between cnn, lstm, and cldnn models in the context of radio modulation classification," in *2020 12th International Conference on Electrical Engineering (ICEENG)*, 2020, pp. 190–195.

[10] A. D. Bodner, A. S. Tepsich, J. N. Spolski, and S. Pourteau, "Convolutional kolmogorov-arnold networks," *arXiv preprint arXiv:2406.13155*, 2024.