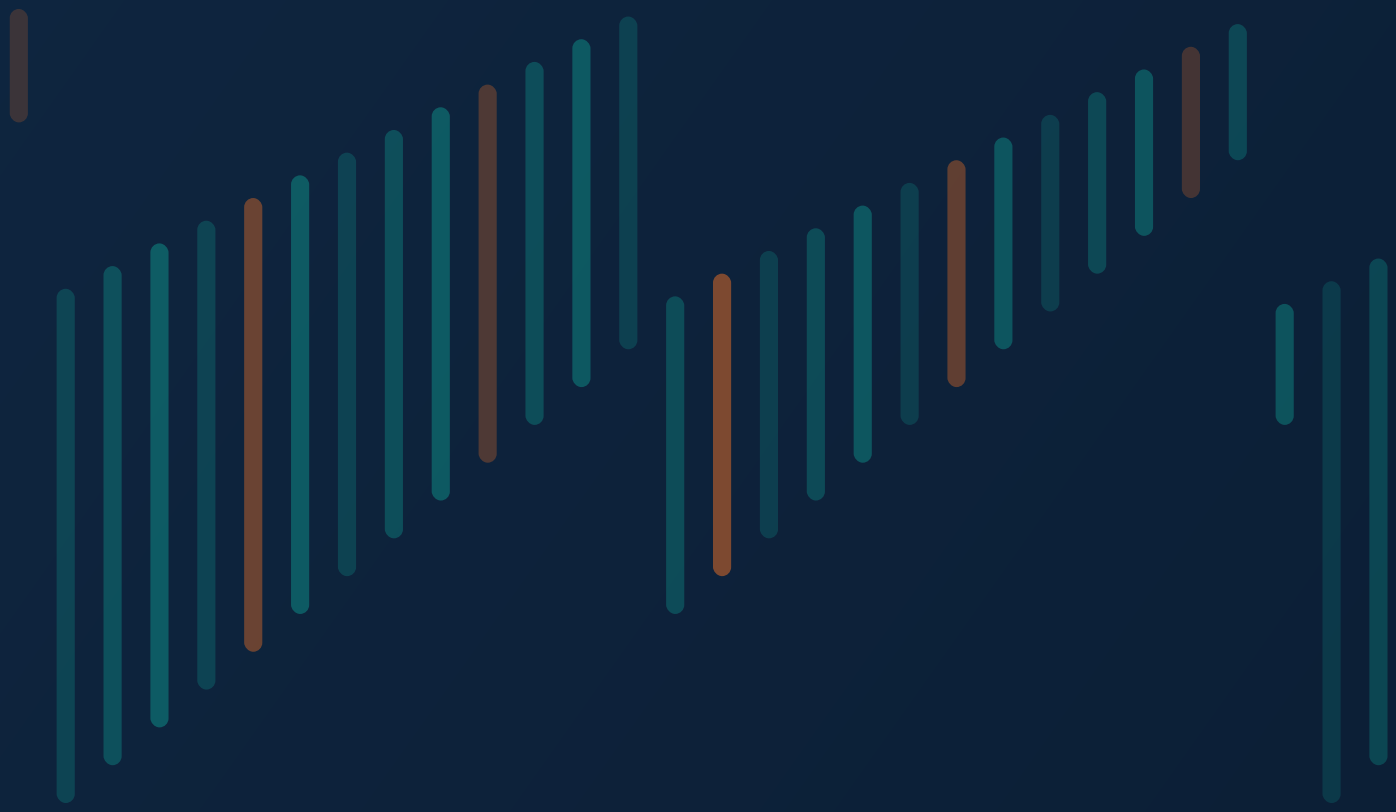
B

भारतीय प्रौद्योगिकी संस्थान जम्मू
INDIAN INSTITUTE OF TECHNOLOGY JAMMU

Identifying seven applications from the shape of their encrypted traffic



B

B

B

B

This is a revised edition of my M.Tech dissertation. The original was submitted to IIT Jammu in September 2022 and approved on 3 October 2022. **The approved 2022 version remains the official record.** This edition does not replace it.

> The text is written from scratch in plain language. The background chapter is much shorter. Statements in the 2022 document that were wrong or unsupported are corrected, and Chapter 7 lists every correction. The reference list is rebuilt so that each citation points to the right work.

> The data, the experiments and every reported number. No experiment was run again. All result figures are reproduced from the 2022 document without editing. One chart in Chapter 5 is new and plots numbers copied from the original comparison table.

I wrote this edition because the original document described the work badly. Its abstract and conclusion claimed things the experiments did not do. The experiments themselves were real and the results are worth reading. This version tries to report them exactly as they are, including the parts that did not work.

The signed approval sheet is not reproduced here because the examiners approved the 2022 text, not this one.

B

Most web traffic is now encrypted. A network operator can no longer open packets to see which application sent them. This work asks how well an application can be identified from flow statistics alone. These are measurements such as packet sizes, timing and direction that stay visible after encryption.

I captured traffic on the IIT Jammu campus network for seven applications. Amazon, Facebook, Instagram, Twitter, Wikipedia, YouTube and Zoom were each used on their own for about 24 hours inside an Ubuntu virtual machine. CICFlowMeter turned the captures into 18,490 flows with 81 numeric features each. I trained five fully connected neural networks with different activation functions, one 1D convolutional network and six tree based ensembles.

Gradient Boosting gave the best result with 96% accuracy and 96% weighted F1. Bagging and XGBoost reached 95%. The 1D CNN reached 87% and the neural network configuration listed in the final comparison reached 83%. YouTube, Instagram and Zoom were recognised almost perfectly. Wikipedia and Facebook stayed weak in every model. Wikipedia precision never went above 68%.

The study has clear limits. IP addresses and port numbers were kept as features, so part of the accuracy may come from server addresses rather than traffic behaviour. Flows were split at random. The confusion matrices show that not every model was evaluated on the same test set. Chapter 7 explains what these limits mean and how a stronger version of the study would be run.

Keywords encrypted traffic classification, flow features, CICFlowMeter, neural networks, gradient boosting, ensemble learning

B B

Encryption hides what people say online. It does not fully hide how their traffic behaves. This work measures how far that behaviour alone can go.

Network traffic classification means looking at traffic and deciding which application or service produced it. Operators use it to plan capacity, give priority to time sensitive traffic such as video calls, measure usage and spot unusual activity that may point to an attack.

A campus network gives a simple example. During an online exam the administrator wants Zoom calls to stay smooth. If the network can recognise Zoom traffic it can give that traffic priority over video streaming. To do that it first has to know which traffic is which.

Port numbers. Every connection is addressed to a port on the receiving machine. Services used to sit on fixed ports, so the port alone named the service. Today almost all web applications use port 443 for secure web traffic, and many use ports that change. The port says very little.

Deep packet inspection. The other old approach reads the contents of each packet and looks for known patterns. Encryption with TLS makes the contents unreadable. Even where inspection is possible it is slow and intrudes on user privacy.

Behaviour. What remains visible is the outside of the traffic. How many packets were sent, how large they were, how much time passed between them and in which direction they travelled. Different applications leave different patterns. Video streaming pulls a heavy steady stream towards the user. A video call sends and receives at a regular rhythm in both directions. Reading an article produces a short burst followed by silence.

I captured my own labelled traffic for seven applications on the campus network. I converted it into flow level statistics with CICFlowMeter and trained twelve classifiers on the same 81 features. I then compared them overall and class by class.

- A labelled dataset of encrypted campus traffic for seven applications, each captured on its own for about 24 hours.
- A comparison of five fully connected networks, one 1D CNN and six tree ensembles on one shared feature set.
- A class level analysis showing where flow statistics succeed and where they fail.
- In this edition, an open account of the study's limits and of the errors in the 2022 document.

This work does not propose a new feature design or a new model. It is an empirical comparison of standard methods on a new dataset.

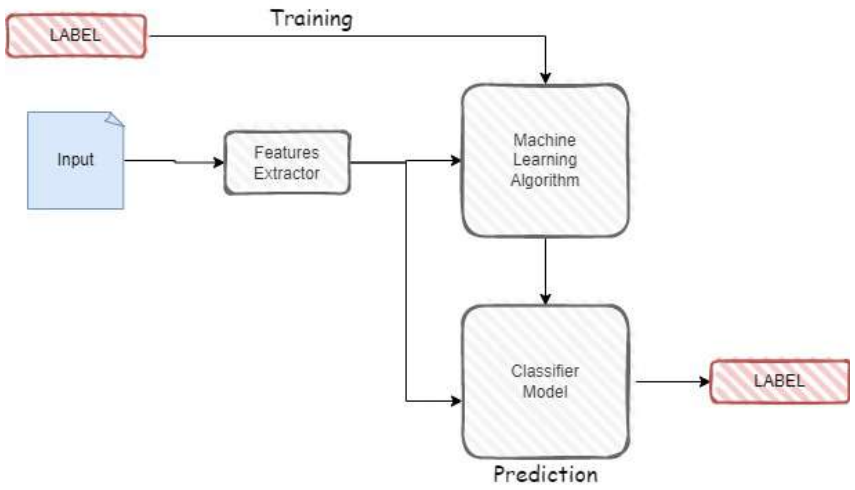
Only the ideas needed to read the results. Readers who know networking and machine learning can skip to Chapter 3.

Data on the internet travels in packets. Each packet carries a header with addresses and control information, followed by the content. TLS, which secures HTTPS, encrypts the content. The headers that routers need stay readable.

A **flow** groups packets that belong to one conversation. It is identified by five values: source IP address, source port, destination IP address, destination port and transport protocol. TCP is the transport protocol for most web pages. It delivers data reliably and in order. UDP is lighter and is often used for live audio and video.

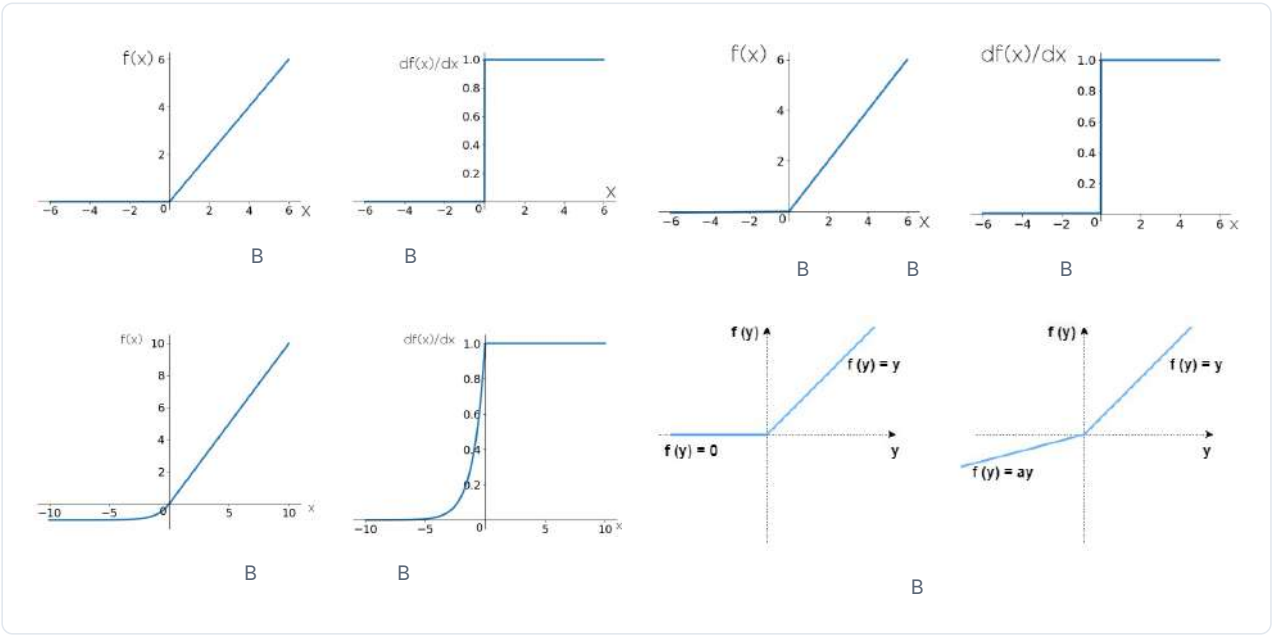
CICFlowMeter, a tool from the Canadian Institute for Cybersecurity, reads packet captures, builds bidirectional flows and computes about eighty statistics for each one [6]. Examples are flow duration, packet counts in each direction, mean and maximum packet length and the time between packets.

This is a supervised classification task. Each example is a flow described by numbers, and each has a known label that names the application. A model learns the link between numbers and labels from training examples and is then judged on examples it has not seen.

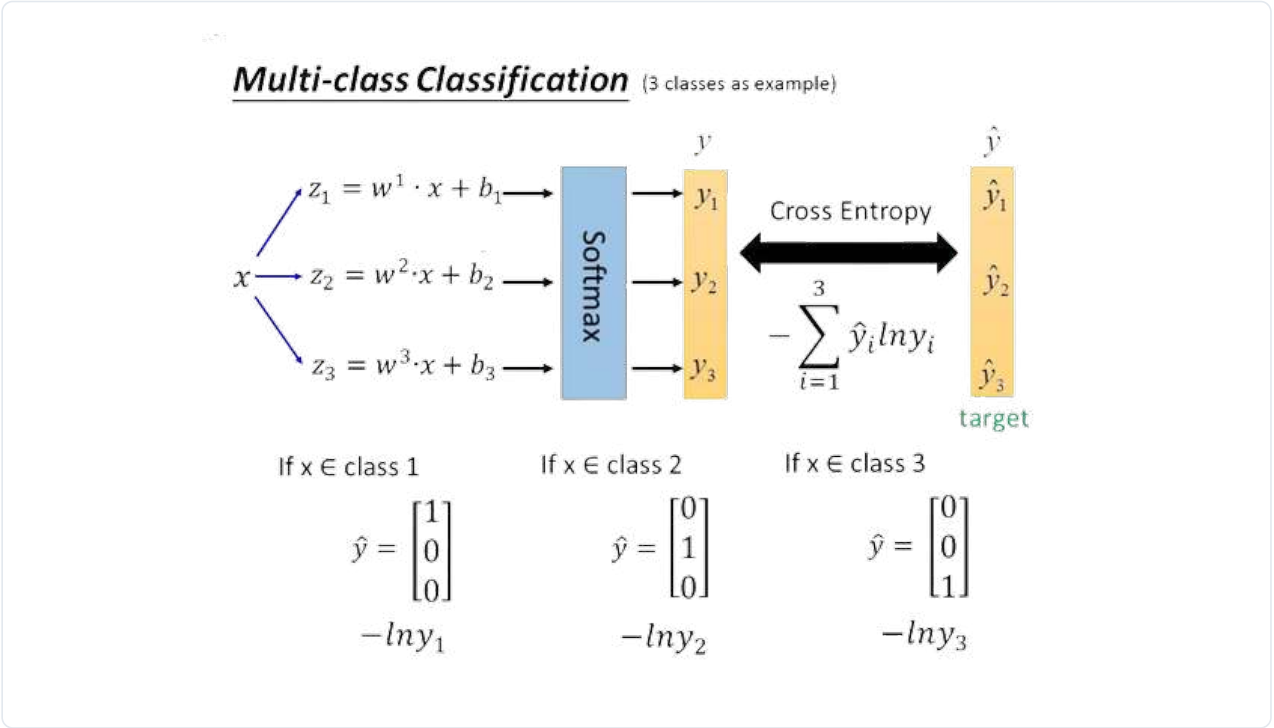


A fully connected neural network passes the input through layers of units. Each unit takes a weighted sum of the previous layer and applies an **activation function**. Without that non linear step a stack of layers could only learn straight line relationships. Training adjusts the weights to reduce a loss that measures how wrong the predictions are.

Four activation functions appear in this work. **ReLU** outputs the input when it is positive and zero otherwise. **Leaky ReLU** lets a small fixed fraction of negative input through. **ELU** uses a smooth exponential curve for negative input. **PReLU** is like Leaky ReLU but learns the size of the negative slope during training [9]. ELU is described in [10].



The last layer has one unit per application and uses **softmax**, which turns the scores into probabilities that add up to one. The loss is **categorical cross entropy**. It is small when the model gives high probability to the right class and large when it is confidently wrong.

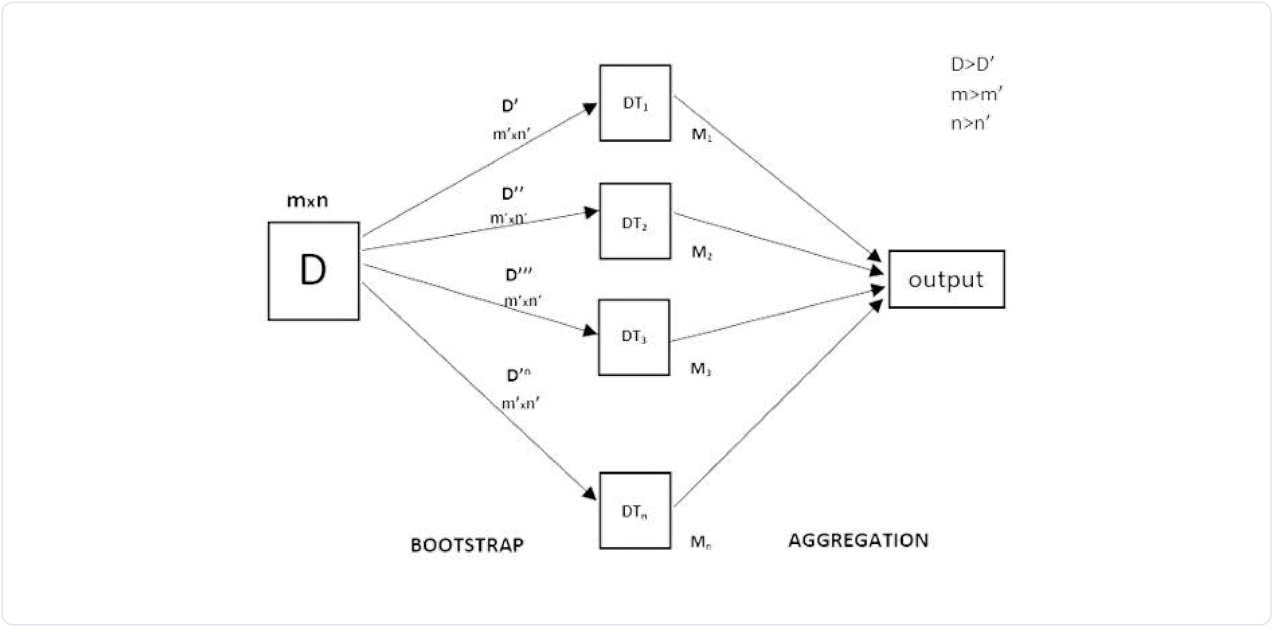


Two tools limit overfitting. **Dropout** switches off a random share of units at each training step so that the network cannot depend on any single unit. It is turned off at test time [7]. **L2 regularisation** adds a penalty for large weights to the loss. The optimiser in all neural models is Adam, a common variant of gradient descent [8].

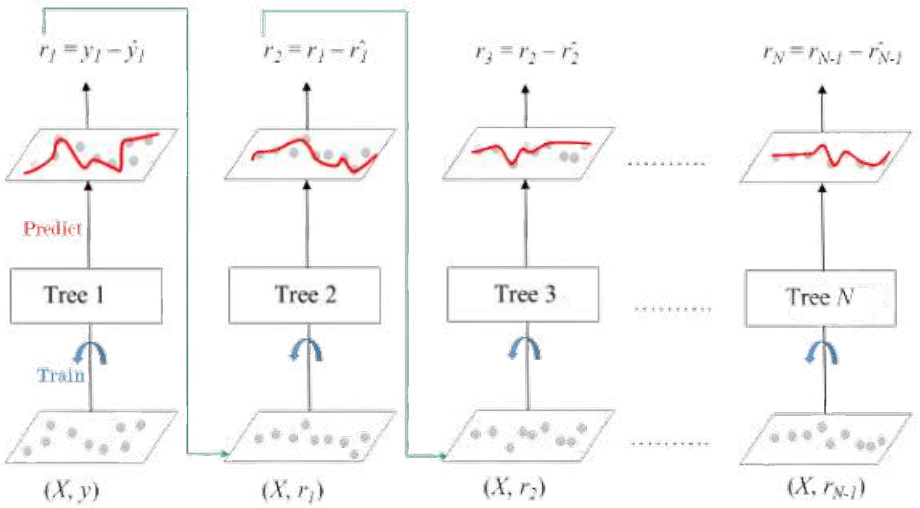
A **one dimensional convolutional network** slides small filters along the input and detects local patterns. It assumes that neighbouring input values are related, which holds for signals and text. Section 6.1 returns to whether it holds for a row of flow statistics.

A **decision tree** classifies an example by asking a series of threshold questions such as whether the mean packet length is above some value. One tree is easy to read but unstable. Ensembles combine many trees.

Bagging trains each tree on a random resample of the training data and takes a majority vote [11]. **Random Forest** adds a second source of randomness by letting each split consider only a random subset of features [12].



Boosting builds trees one after another. **AdaBoost** gives more weight to examples that earlier trees got wrong [13]. **Gradient Boosting** trains each new tree on the remaining error of the trees before it and adds its output scaled by a learning rate [14]. **XGBoost** is a fast and widely used implementation of gradient boosted trees with extra regularisation [15]. **Stacking** trains several different base models and then a second model that learns how to combine their predictions. Here the second model is logistic regression [16].



Classifying traffic from flow statistics with machine learning has a long history. Moore and Zuev applied Bayesian methods to flow features in 2005 [1]. Draper-Gil and colleagues released the ISCX VPN-nonVPN dataset and classified encrypted and VPN traffic using time related features [2]. Later work moved towards deep learning on raw packets or packet sequences. Deep Packet used convolutional networks and stacked autoencoders on packet bytes [3]. Aceto and colleagues studied deep learning for mobile app traffic [4]. Akbari and colleagues argued that many deep models depend on handshake fields that newer protocols hide and proposed features that generalise better [5].

This thesis sits at the simple end of that range. It uses standard CICFlowMeter features and standard models. Its value lies in a new labelled dataset and a careful look at per class behaviour, not in a new method.

Seven applications. Twenty four hours each. One application running at a time.

All traffic was captured on the IIT Jammu campus network inside a virtual machine. During each capture only the application under study was in use. Every other application was closed so that the captured traffic could be labelled by the application that was running.

.	
	B
	B

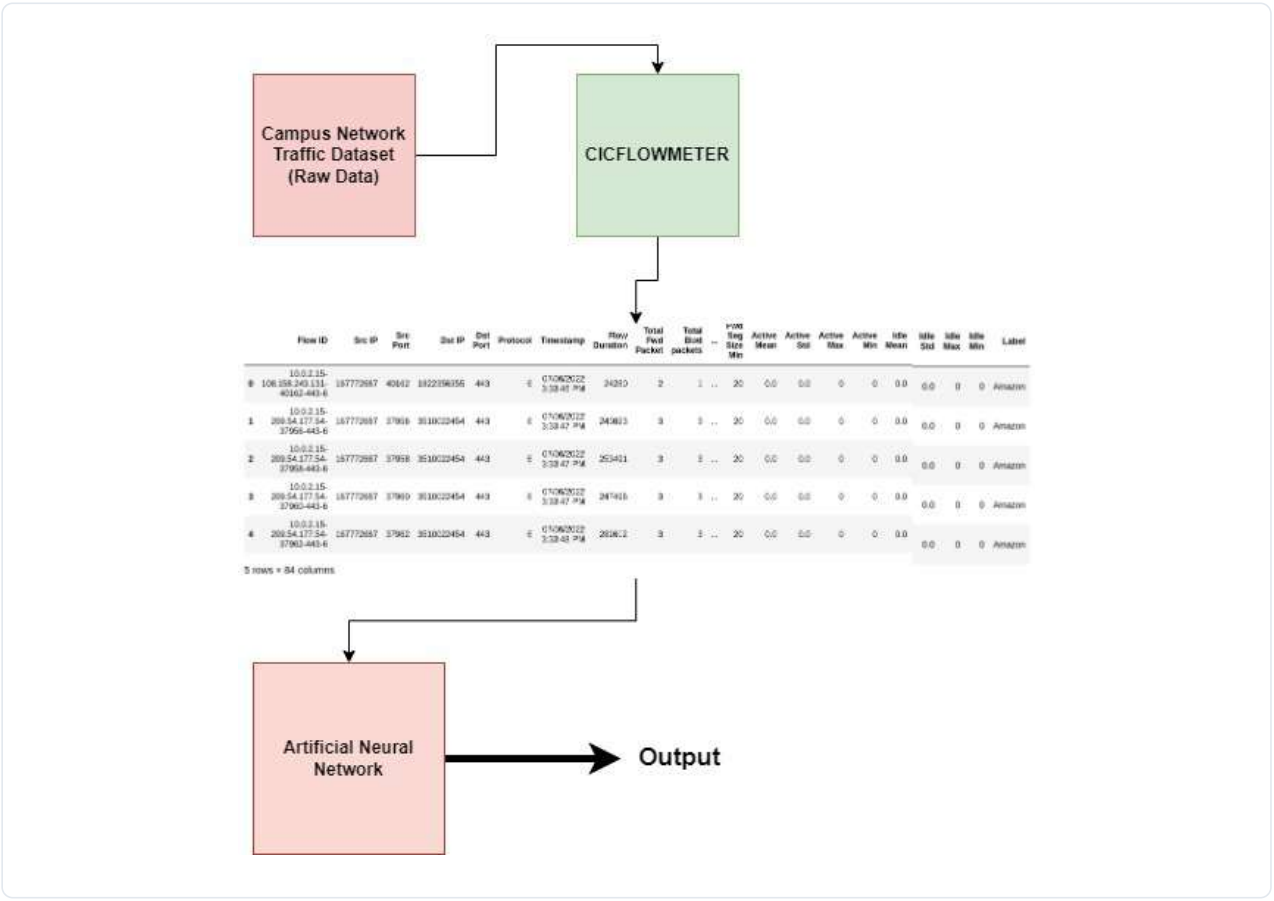
B B B B B

B
B

B

Wikipedia use was the narrowest of the seven. Reading articles creates short bursts of traffic and long pauses. In some test sets this class ended up with far fewer flows than the others, which matters for the results.

Each application's capture was passed through CICFlowMeter separately and every resulting flow was labelled with that application. The combined table had **18,494 rows and 84 columns**.



1. Rows with missing values were removed. This left **18,490 rows**.
2. IP addresses were converted from dotted text to integers using Python's `socket.inet_aton` and `struct.unpack`.
3. Flow ID and Timestamp were dropped. Label was separated out as the target.
4. The seven labels were encoded as integers.
5. The features were normalised. The 2022 document does not say which normalisation method was used.

This left **81 input features** for every model.

B B B B B

B

The 81 features include the source and destination IP addresses, the source and destination ports and the protocol. Only Flow ID, Timestamp and Label were removed. The 2022 conclusion said that IP addresses and ports were masked. That statement was wrong. Because one application tends to talk to the same servers, a model can learn server addresses instead of traffic behaviour. Section 7.2 discusses what this means for the results.

For the mixed activation network, the CNN and all ensembles the 2022 document states the split. 15% of the data was held out for testing, and 15% of the remainder was used for validation. The split was random at the level of single flows. The split for the first four neural networks is not stated.

The confusion matrices in Chapter 5 reveal more than the text does. Adding up their rows gives the number of test flows per class.

B

B

B

Two things follow. First, both test sets are larger than 15% of 18,490 flows, which is about 2,774. Second, the class mix is very different. In the first set Wikipedia is under 2% of the test flows. In the second set every class is close to one seventh. The 2022 document does not explain this. The data may have been resampled before some runs, but nothing in the document records it. What can be said is that **the twelve models were not all evaluated on the same test set**, so small differences between them should not be over read. No confusion matrices were saved for the ensembles, so their test set cannot be checked.

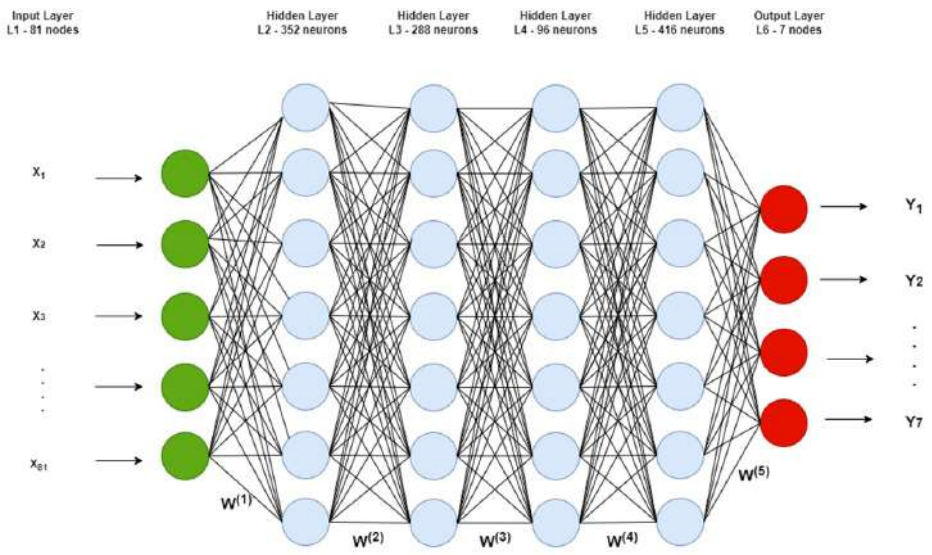
Twelve classifiers trained on the same 81 features. All settings below are taken from the 2022 document.

All neural networks were built with the Keras Sequential API. Each takes the 81 features as input and ends in a softmax layer with seven units. Each was trained for 200 epochs with the Adam optimiser and categorical cross entropy loss. Layer sizes were chosen by random search. The search ranges, random seeds and library versions were not recorded.

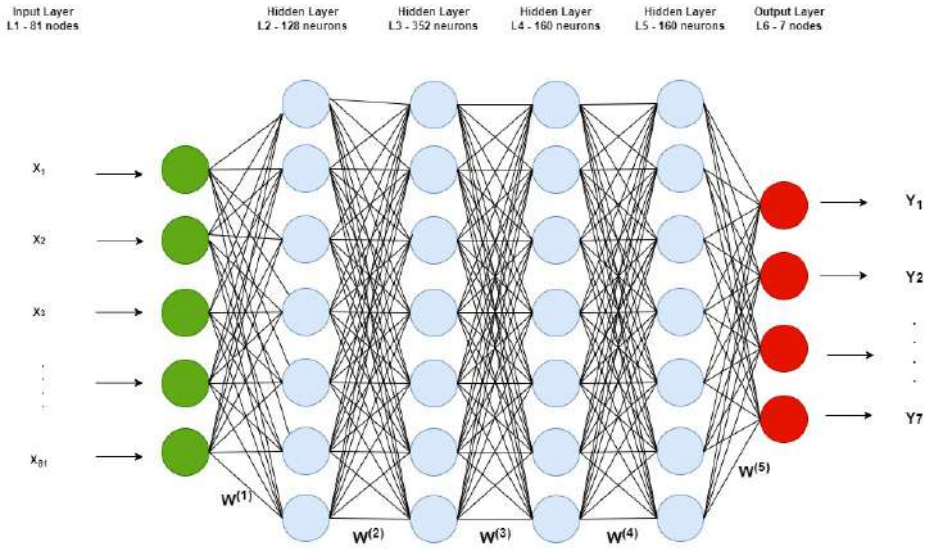
B
B
B
B
B
B

B B B

The first four networks were trained in sequence, each one a response to the training curves of the one before. The 2022 document read those curves as underfitting. Section 6.3 explains why that reading was mostly wrong. The mixed network added L2 regularisation and either lowered dropout to 0.03 or removed it. The 2022 text mentions both and does not say which run produced the reported results.



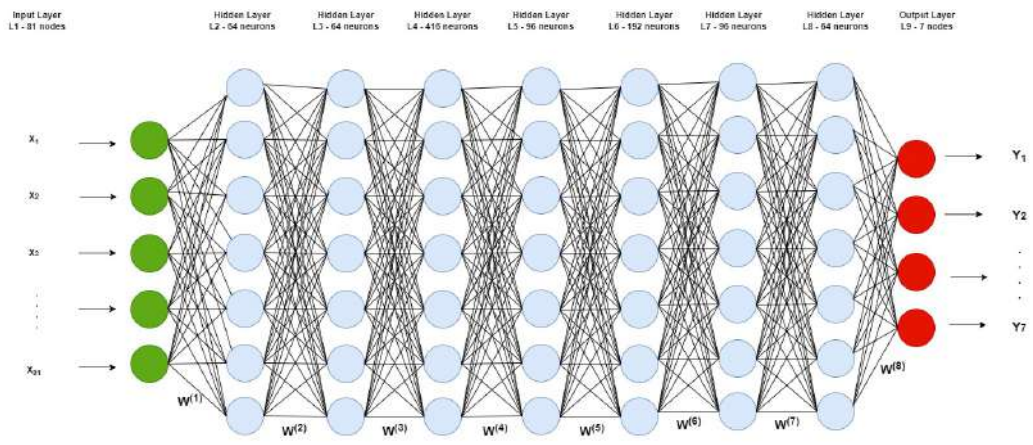
B



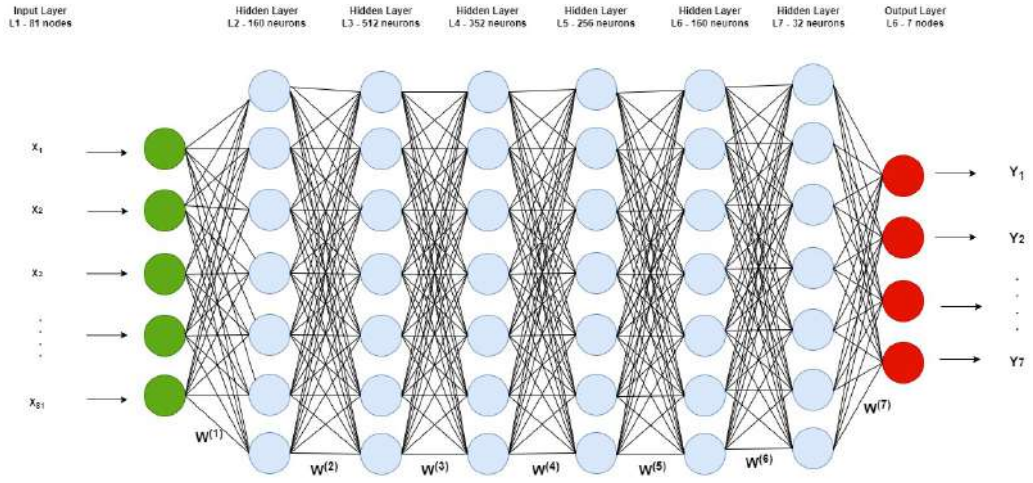
B

B

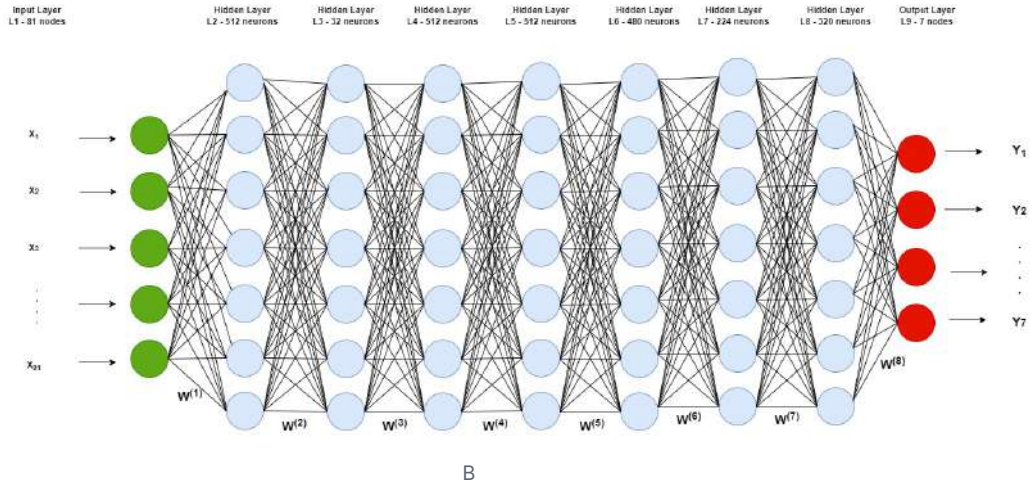
B B B


Input Layer
L1 - 81 nodes
Hidden Layer
L2 - 64 neurons
Hidden Layer
L3 - 64 neurons
Hidden Layer
L4 - 416 neurons
Hidden Layer
L5 - 96 neurons
Hidden Layer
L6 - 152 neurons
Hidden Layer
L7 - 96 neurons
Hidden Layer
L8 - 64 neurons
Output Layer
L9 - 7 nodes
X1
X2
X3
X81
Y1
Y2
Y7
W(1)
W(2)
W(3)
W(4)
W(5)
W(6)
W(7)
W(8)

B B


Input Layer
L1 - 81 nodes
Hidden Layer
L2 - 160 neurons
Hidden Layer
L3 - 512 neurons
Hidden Layer
L4 - 352 neurons
Hidden Layer
L5 - 256 neurons
Hidden Layer
L6 - 160 neurons
Hidden Layer
L7 - 32 neurons
Output Layer
L8 - 7 nodes
X1
X2
X3
X81
Y1
Y2
Y7
W(1)
W(2)
W(3)
W(4)
W(5)
W(6)
W(7)

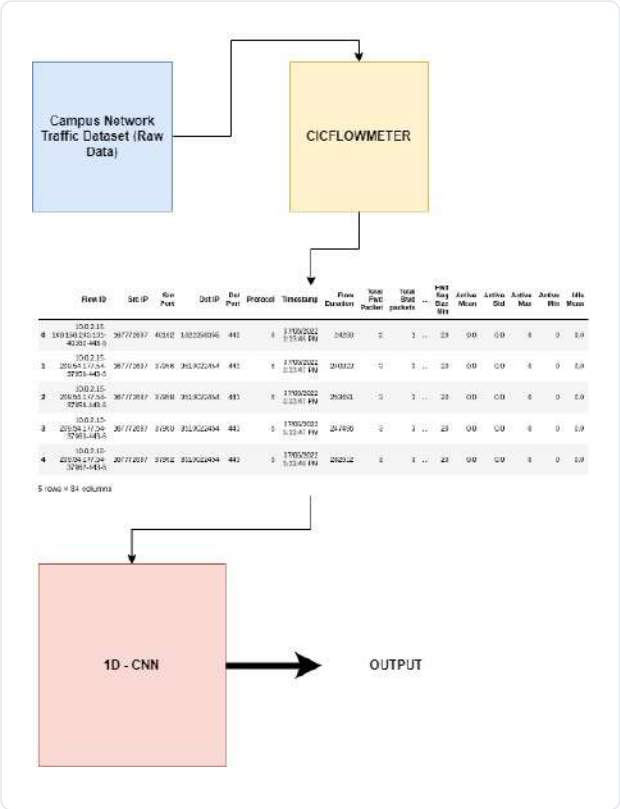
B

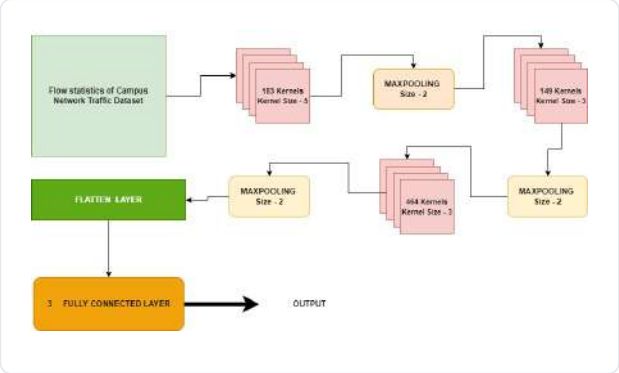

Input Layer
L1 - 81 nodes
Hidden Layer
L2 - 512 neurons
Hidden Layer
L3 - 32 neurons
Hidden Layer
L4 - 512 neurons
Hidden Layer
L5 - 512 neurons
Hidden Layer
L6 - 480 neurons
Hidden Layer
L7 - 224 neurons
Hidden Layer
L8 - 320 neurons
Output Layer
L9 - 7 nodes
X1
X2
X3
X81
Y1
Y2
Y7
W(1)
W(2)
W(3)
W(4)
W(5)
W(6)
W(7)
W(8)

B

B

The CNN has three convolution layers with 183, 149 and 464 filters and kernel sizes of 5, 3 and 3. Each is followed by ReLU and max pooling of size 2. The output is flattened and passed to dense layers of 96, 192 and 352 units with ReLU, then to the seven unit softmax output. L2 regularisation with alpha 0.1 was used. The 2022 text gives the learning rate and decay as "103" and "106". These are almost certainly 10^{-3} and 10^{-6} with the minus signs lost in typesetting.





B B B

.

B .

.

.

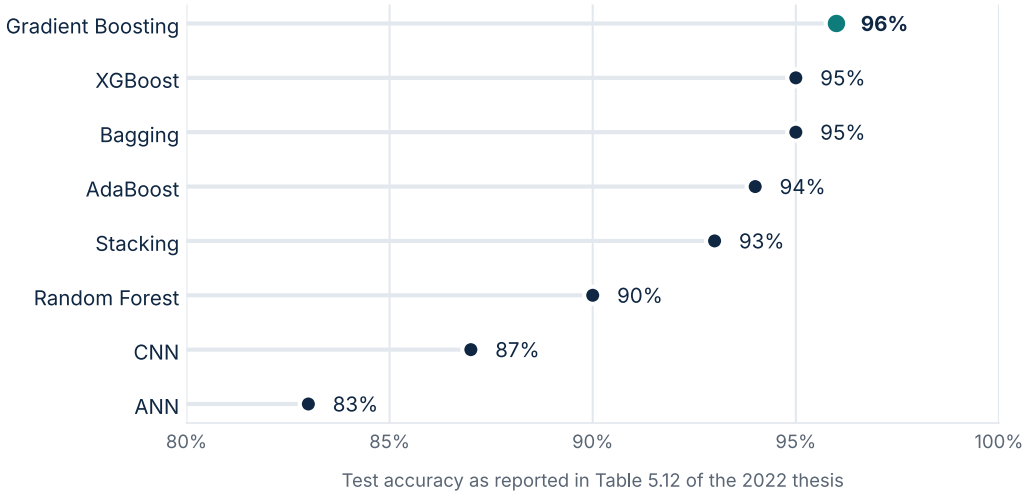
For each class, **precision** asks how often the model is right when it predicts that class. **Recall** asks how many of the true examples of that class it finds. **F1** is the harmonic mean of the two. **Accuracy** is the share of all test flows labelled correctly.

The weighted average weights each class by its number of test flows. Weighted recall is therefore equal to accuracy. Large classes dominate both numbers, which is why a model can post a high weighted score while failing on a small class.

Every number in this chapter is copied from the 2022 thesis without change.

	B
B	
.	
B .	
.	

B B B



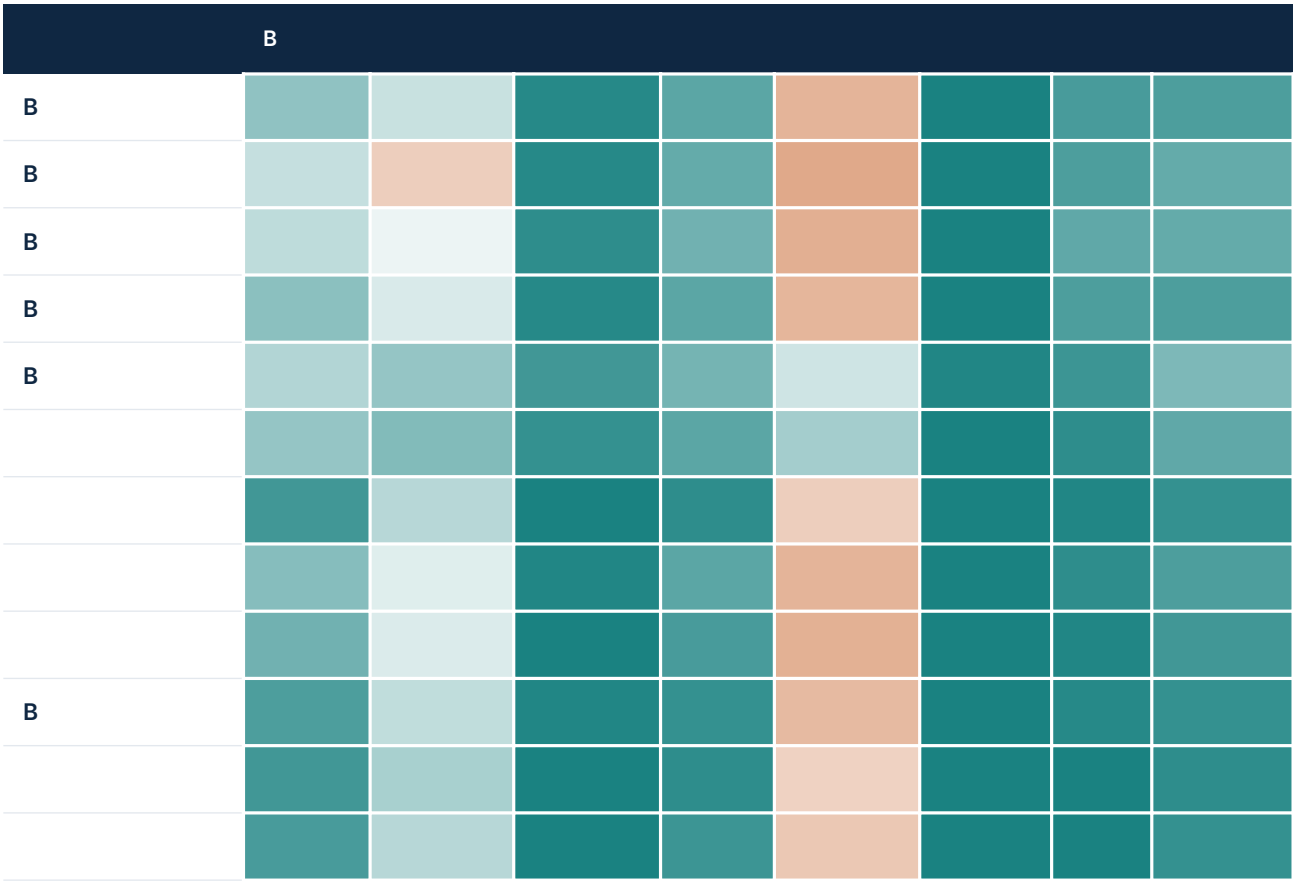
The tree ensembles take the top five places. Gradient Boosting leads at 96%, followed by Bagging and XGBoost at 95% and AdaBoost at 94%. Random Forest is the weakest ensemble at 90%, possibly because its trees were limited to depth 7. The CNN reaches 87%.

B B

The ANN row is the mixed activation network. The ReLU and PReLU networks reached higher weighted scores, 91% F1 with 90% and 89% weighted recall. Those scores came from the unbalanced test set described in Section 3.4, where large easy classes dominate. They cannot be compared directly with the mixed network, which was tested on a balanced set. The 2022 conclusion gave the ANN as 92%. No table in the document supports that number.

Overall numbers hide most of the story. The table below shows F1 for every class and every model. Teal shading deepens as F1 rises. Orange marks values below 50%.

B B B

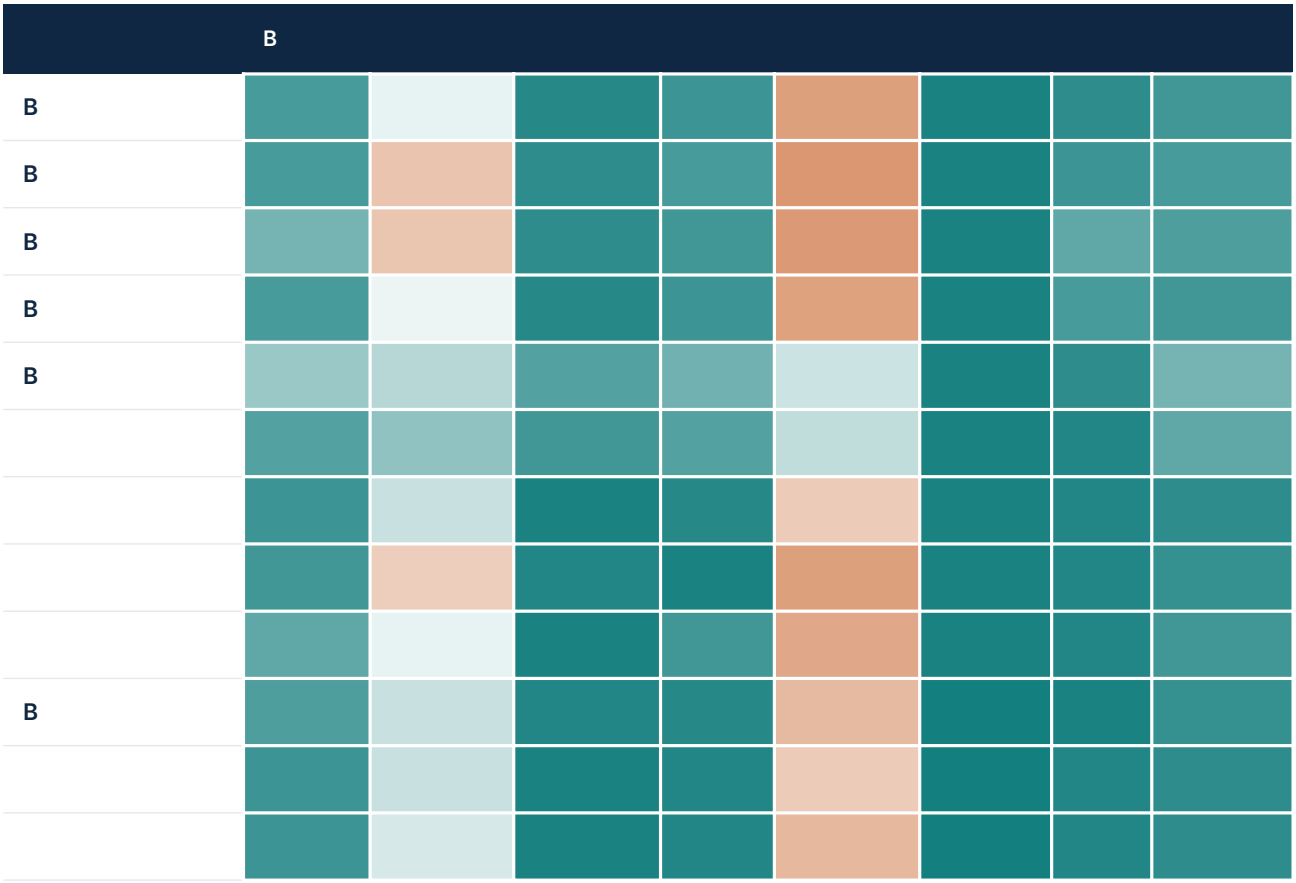


B B

Three patterns stand out.

- **YouTube, Instagram and Zoom are easy.** YouTube scores 98 or 99 in every model. Instagram never drops below 93. Zoom stays at 88 or above.
- **Amazon and Twitter are good but not perfect.** Bagging and the three boosting models do clearly better than neural networks on Amazon, mostly through higher recall.
- **Wikipedia and Facebook are hard everywhere.** Wikipedia F1 runs from 25 to 75. Facebook runs from 45 to 82.

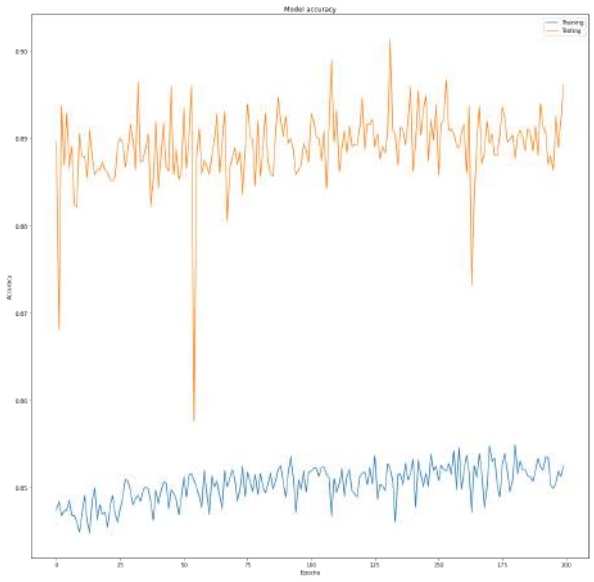
Wikipedia precision is the weakest number in the study.

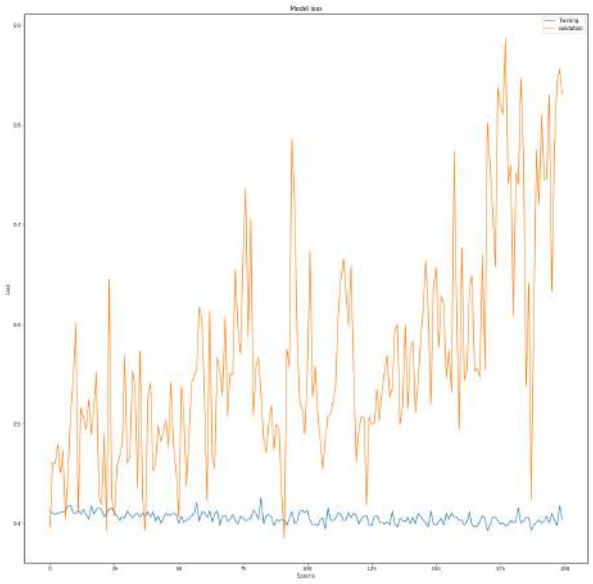


One value in the original tables does not add up. AdaBoost on Facebook shows precision 66% and recall 75%, which gives an F1 of about 70%. The table reports 68%. It is reproduced unchanged.

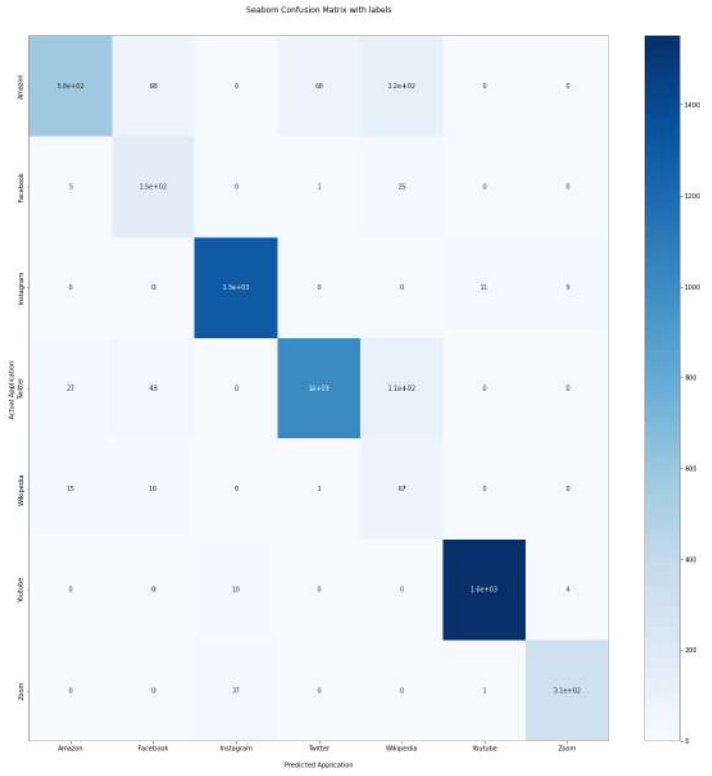
Each figure below shows the accuracy and loss curves over 200 epochs and the confusion matrix on the held out set. In the matrices, rows are true applications and columns are predicted ones. Correct predictions lie on the diagonal. In the first four figures the curve labelled Testing in the accuracy plot and the curve labelled validation in the loss plot appear to be the same held out data. The 2022 document does not say which set they are.

B B B


Model accuracy
Training
Testing
Accuracy
Epochs


Model loss
Training
validation
Loss
Epochs

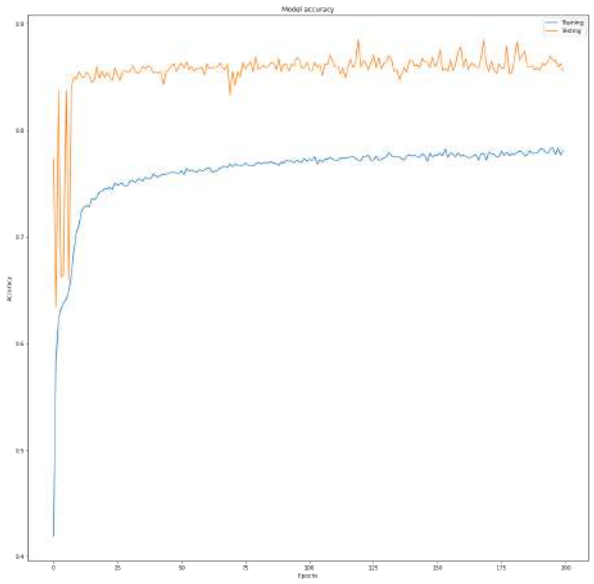
B B

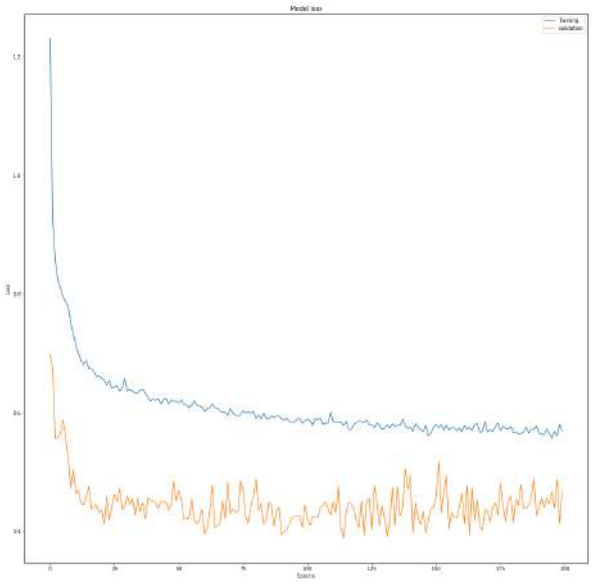

Seaborn Confusion Matrix with labels
Actual Application
Amazon
Facebook
Instagram
Twitter
Wikipedia
Youtube
Zoom
Predicted Application

B

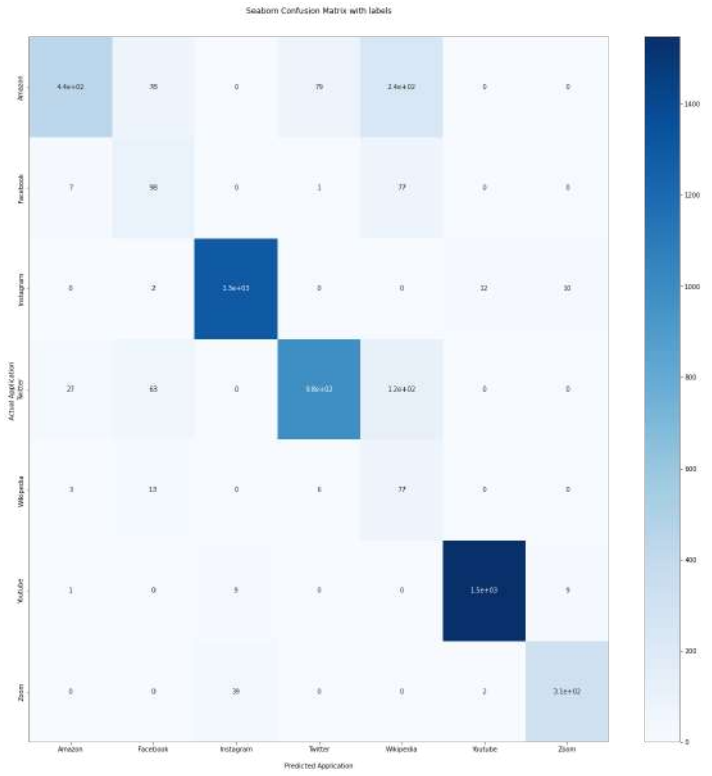
B

B B B


Model accuracy
Training
Testing
Accuracy
Epochs


Model loss
Training
validation
Loss
Epochs

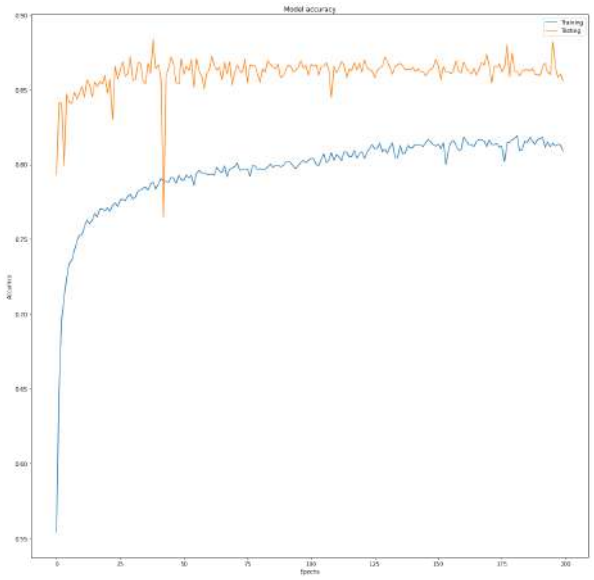
B B

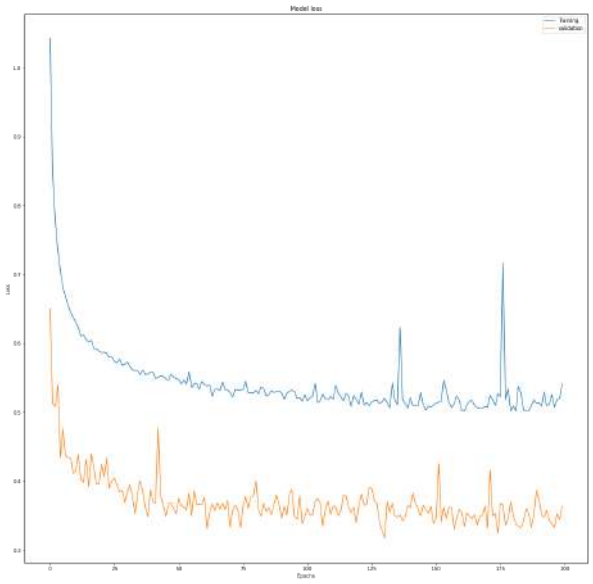

Seaborn Confusion Matrix with labels
Actual Application
Amazon
Facebook
Instagram
Twitter
Wikipedia
Youtube
Zoom
Predicted Application
Amazon
Facebook
Instagram
Twitter
Wikipedia
Youtube
Zoom

B

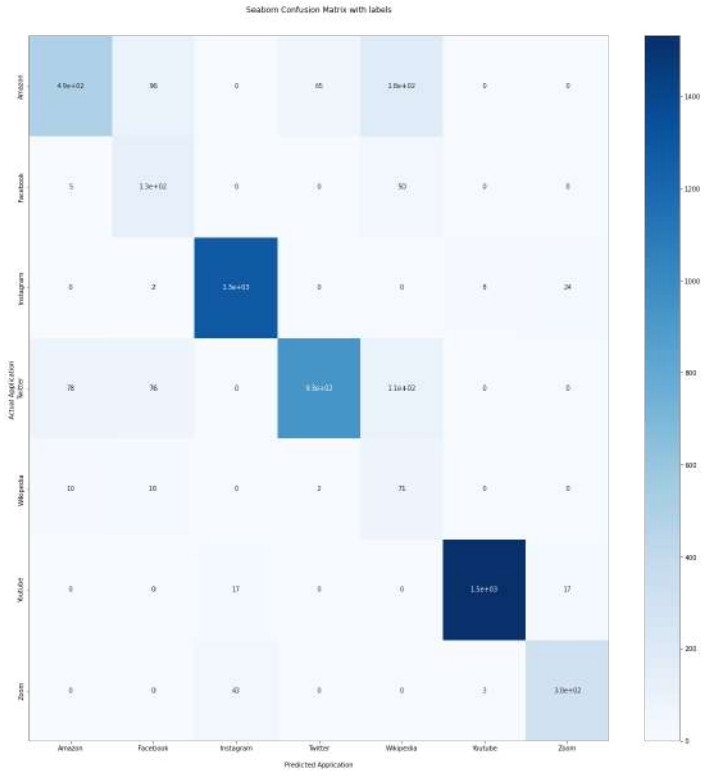
B

B B B


Model accuracy
Training
Testing
Accuracy
Epochs


Model loss
Training
validation
Loss
Epochs

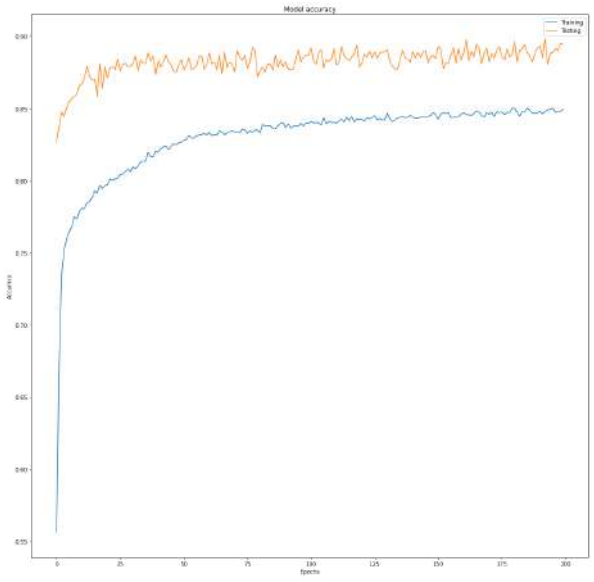
B B

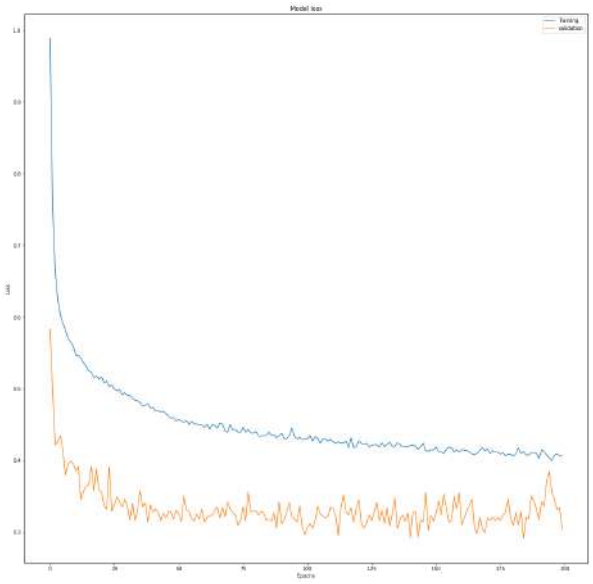

Seaborn Confusion Matrix with labels
Actual Application
Amazon
Facebook
Instagram
Twitter
Wikipedia
Youtube
Zoom
Predicted Application

B

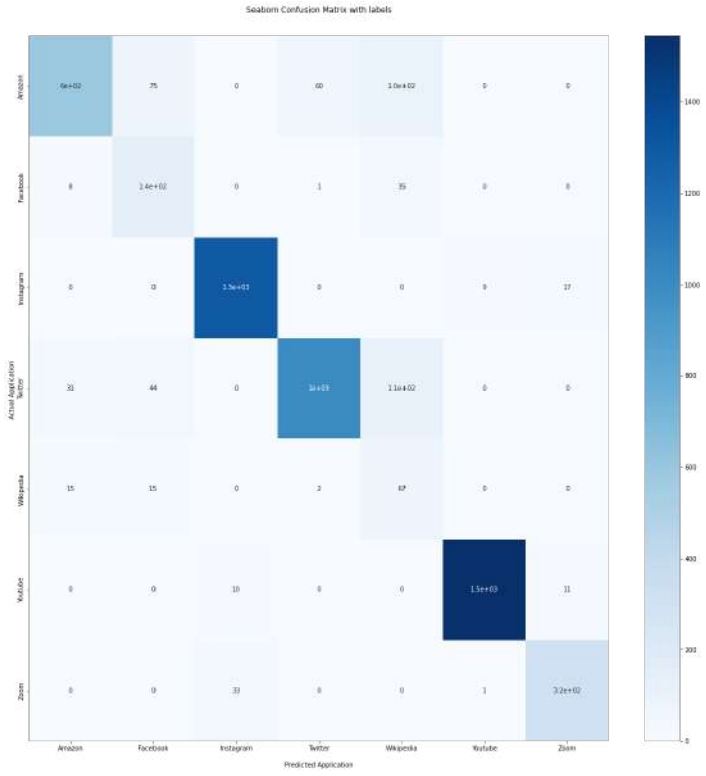
B

B B B


Model accuracy
Training
Testing
Accuracy
Epochs

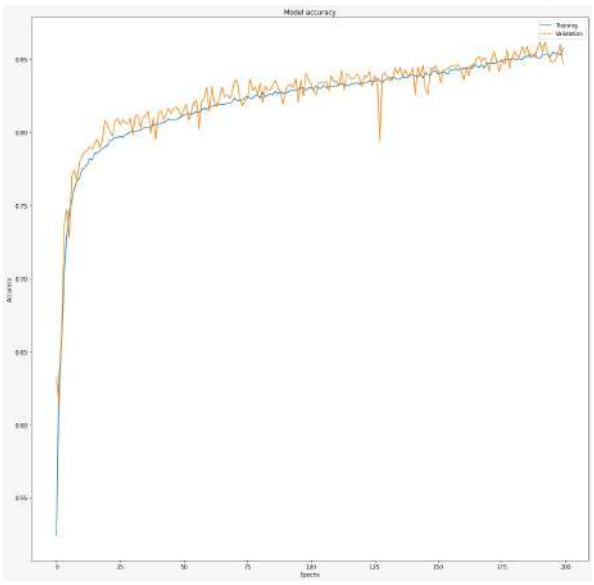

Model loss
Training
validation
Loss
Epochs

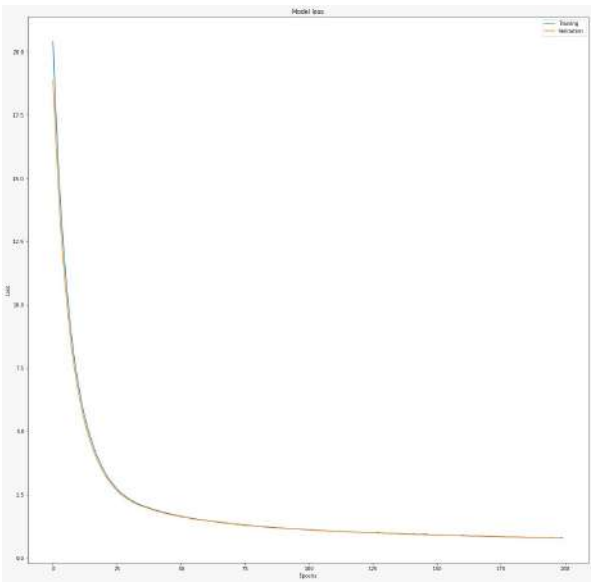
B B


Seaborn Confusion Matrix with labels
Actual Application
Amazon
Facebook
Instagram
Twitter
Wikipedia
Youtube
Zoom
Predicted Application

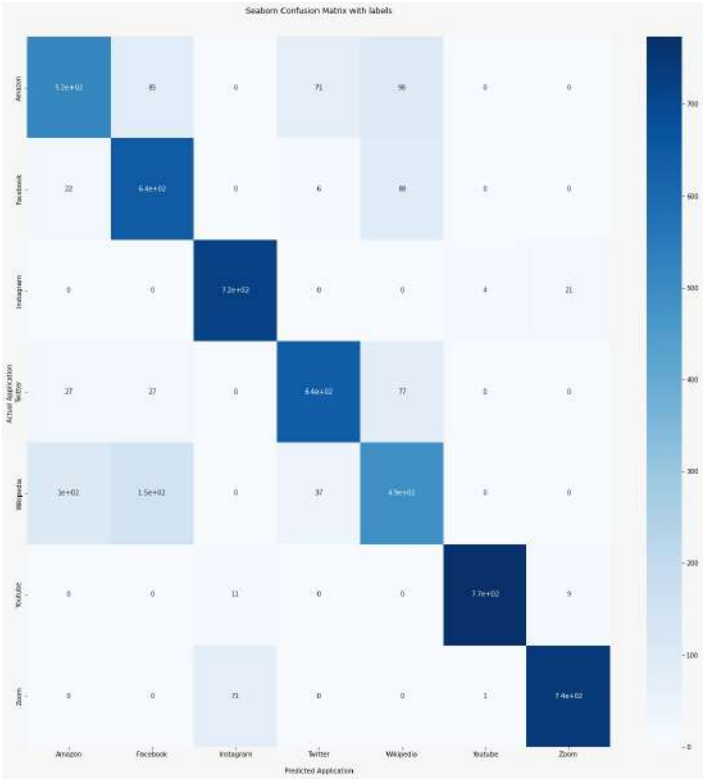
B

B


Model accuracy
Training
Validation
Accuracy
Epochs


Model loss
Training
Validation
Loss
Epochs

B B

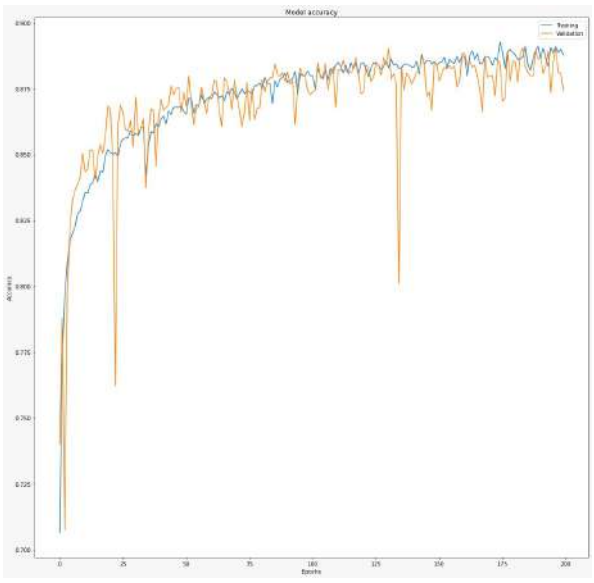

Seaborn Confusion Matrix with labels
Amazon
Facebook
Instagram
Twitter
Wikipedia
Youtube
Zoom
Actual Application
Predicted Application

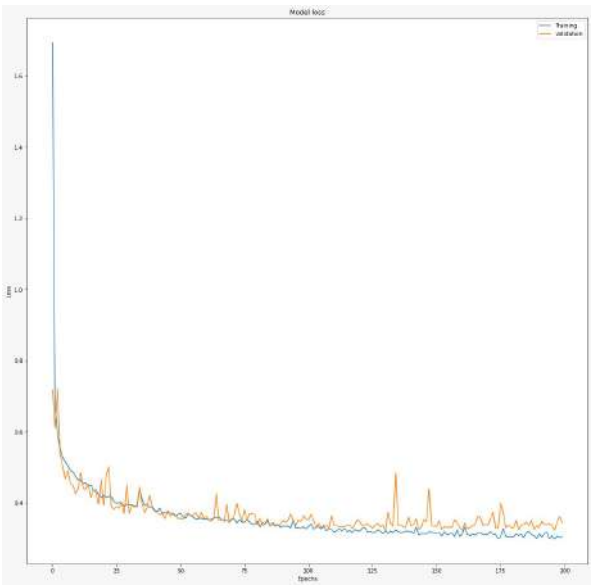
B

B

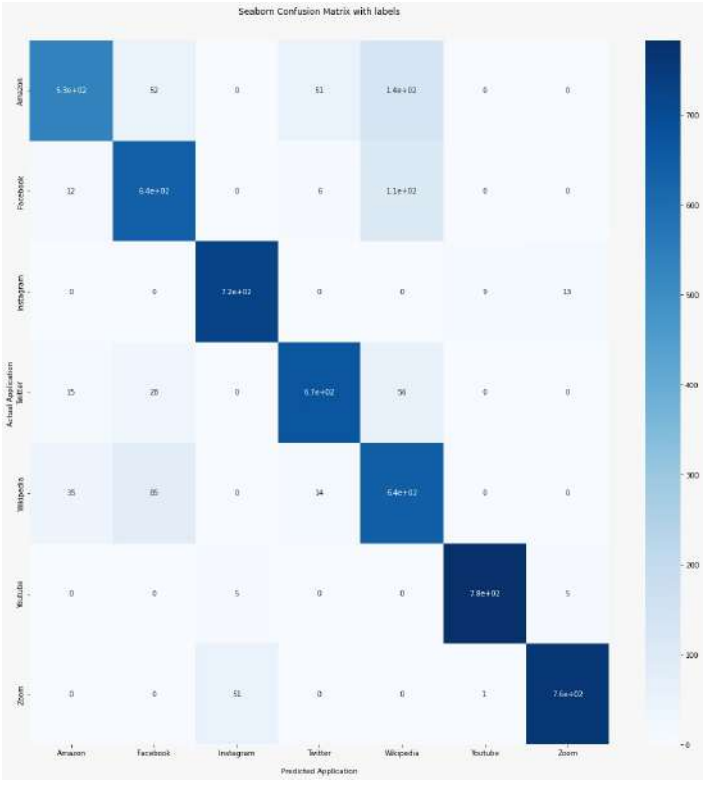
B

B B B





B B



B

The ensembles were evaluated with classification reports only. No curves or confusion matrices were saved for them.

Trees suited this data. Some applications have a clear signature and some do not. And the training curves meant something different from what the 2022 text said.

The input is a table. Each column is a separate measurement with its own meaning and scale. Decision trees work directly with thresholds on single columns, which fits this kind of data well. Later work by Grinsztajn and colleagues found the same pattern across many tabular benchmarks [18].

The CNN assumes that neighbouring inputs are related. In a row of CICFlowMeter features the order of columns is arbitrary, so the sliding filters have little real structure to find. The CNN still did better than the fully connected networks here, but this study cannot say why.

A further reason may be the IP address and port columns. A tree can split on a destination address in a single step. If those columns carry much of the signal, trees would gain the most from them. This was not tested.

YouTube, Instagram and Zoom each have a distinct traffic shape. Streaming video is heavy and one sided. A video call is steady and two sided. Instagram loads a lot of media in bursts. Wikipedia, Facebook, Amazon and Twitter all look more like ordinary web browsing, with page loads of text and images followed by pauses. Their flows overlap, and the confusion matrices show them mixing with each other.

Wikipedia also suffered from its size in the unbalanced test set. In the ReLU network about 100 test flows were Wikipedia. The model labelled 322 flows as Wikipedia and 67 of them were correct, which is the reported 20% precision. Most of the wrong ones were Amazon and Twitter flows.

When a class is rare, even a small error rate in large classes swamps it. In the balanced test sets of the mixed network and the CNN, Wikipedia precision rose to 65% and 68%. Part of that rise comes from the class mix rather than from a better model.

> I

The 2022 document called the first four networks underfit because their training accuracy was below their held out accuracy. That pattern is expected with dropout of 0.5. Dropout switches off half the units during training, which lowers training accuracy, and is turned off at evaluation. The gap says more about how the numbers were measured than about the model.

The ReLU network shows something else. Its held out loss rises and becomes unstable in later epochs while held out accuracy stays roughly level. This usually means the model is becoming overconfident on the examples it gets wrong. That is closer to overfitting or unstable training than to underfitting.

Lowering or removing dropout and adding L2 in the mixed network brought the training and validation curves together. That change came together with a change in test set, so the curves and the final scores of the mixed network cannot be compared directly with the first four networks.

What the 2022 document got wrong, what the study cannot show, and how I would run it today.

B B B B

B

B B

B

.

B

.

B B

.

B

Server addresses as features. Keeping IP addresses and ports lets a model learn which servers an application talks to. That works inside one capture but breaks when servers change or when the same model is used on another network. The reported accuracy is probably an upper bound on what traffic behaviour alone can achieve. Arp and colleagues list this kind of shortcut among the common pitfalls of machine learning in security [17].

Random split of flows. Flows from the same browsing session look alike. A random split places close relatives of training flows in the test set, which also raises scores.

Different test sets. As shown in Section 3.4, not all models were tested on the same data. Small differences in Table 5.1 should not be read as real differences between models.

No record of variability. Each model was run once. No seeds, versions or confidence intervals were recorded and no significance test was done. The one point gap between Gradient Boosting and Bagging may not be meaningful.

Narrow data. One machine, one browser, one network and seven applications. Background traffic from the operating system may also have been captured and labelled as the running application.

Protocol versions. The captures were not split by TLS 1.3 [20] or QUIC [21]. These protocols hide more of the handshake, and results may differ under them.

1. Remove IP addresses, ports and other identifiers from the features and report how much accuracy changes.
2. Split by time, training on earlier hours of each capture and testing on later hours.
3. Fix one test set and evaluate every model on it, with class counts reported.
4. Record seeds and versions, repeat each run several times and report the mean and spread.
5. Save confusion matrices for every model, including the ensembles.
6. Filter operating system background traffic before labelling.
7. Report which features the best models rely on.

Flow statistics can identify some encrypted applications very well and others poorly. Reporting both is the result.

This thesis compared twelve classifiers for identifying seven applications from encrypted campus traffic using 81 CICFlowMeter flow features. Tree ensembles worked best. Gradient Boosting reached 96% accuracy, Bagging and XGBoost 95%, and the 1D CNN 87%.

The per class results matter more than the headline. YouTube, Instagram and Zoom were recognised almost perfectly. Wikipedia and Facebook stayed weak in every model, and Wikipedia precision never rose above 68%. Applications whose traffic looks like ordinary browsing are hard to separate with flow statistics alone.

The headline numbers should be read with care. IP addresses and ports were part of the features, flows were split at random and not every model was tested on the same set. A cleaner study would likely report lower scores and a fairer comparison.

- **A cleaner rerun** following Section 7.3, starting with the removal of address features.
- **Robustness to evasion.** An application could pad packets or change its timing to look like another one. Realistic attacks must respect what real traffic can look like, as Pierazzi and colleagues argue [19]. Testing these classifiers against such shaping is a natural next step.
- **Newer protocols** such as TLS 1.3 and QUIC, which hide more metadata.
- **Unknown applications.** Real networks carry traffic from applications the model never saw. A deployed classifier must recognise that it does not know.
- **Deployment cost.** Speed and memory use when classifying live traffic.

B B

Precision, recall and F1 for every class, copied from original Tables 5.1 to 5.11. Values below 50% F1 are marked.

B

B

B

B

B

B

B

B

B

B

B

B

B

B

B

B

B

B

Only works discussed in the text. Works marked with an asterisk were not cited in the 2022 version.

A. W. Moore and D. Zuev. Internet traffic classification using Bayesian analysis techniques. ACM SIGMETRICS Performance Evaluation Review 33(1), 50-60, 2005.

G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun and A. A. Ghorbani. Characterization of encrypted and VPN traffic using time-related features. ICISSP 2016.

M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade and M. Saberian. Deep Packet: A novel approach for encrypted traffic classification using deep learning. Soft Computing 24, 2020.

G. Aceto, D. Ciuonzo, A. Montieri and A. Pescapè. Mobile encrypted traffic classification using deep learning. Network Traffic Measurement and Analysis Conference (TMA) 2018.

I. Akbari, M. A. Salahuddin, L. Ven, N. Limam, R. Boutaba, B. Mathieu, S. Moteau and S. Tuffin. A look behind the curtain: Traffic classification in an increasingly encrypted web. Proceedings of the ACM on Measurement and Analysis of Computing Systems 5(1), 2021.

A. H. Lashkari et al. CICFlowMeter network traffic flow generator. Canadian Institute for Cybersecurity, University of New Brunswick.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929-1958, 2014.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. ICLR 2015.

K. He, X. Zhang, S. Ren and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. ICCV 2015.

D.-A. Clevert, T. Unterthiner and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). ICLR 2016.

L. Breiman. Bagging predictors. Machine Learning 24(2), 123-140, 1996.

L. Breiman. Random forests. Machine Learning 45(1), 5-32, 2001.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119-139, 1997.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29(5), 1189-1232, 2001.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. KDD 2016.

D. H. Wolpert. Stacked generalization. Neural Networks 5(2), 241-259, 1992.

D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro and K. Rieck. Dos and don'ts of machine learning in computer security. USENIX Security 2022.

L. Grinsztajn, E. Oyallon and G. Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? NeurIPS 2022 Datasets and Benchmarks Track.

F. Pierazzi, F. Pendlebury, J. Cortellazzi and L. Cavallaro. Intriguing properties of adversarial ML attacks in the problem space. IEEE Symposium on Security and Privacy 2020.

E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.

J. Iyengar and M. Thomson. QUIC: A UDP-based multiplexed and secure transport. RFC 9000, 2021.