THE UNIVERSITY of York

RTS York

Scheduling IDK Classifiers with Arbitrary Dependences to Minimize the Expected Time to Successful Classification

Tarek Abdelzaher, Kunal Agrawal, Sanjoy Baruah, Alan Burns,
Robert I. Davis, Zhishan Guo, Yigong Hu

Overview

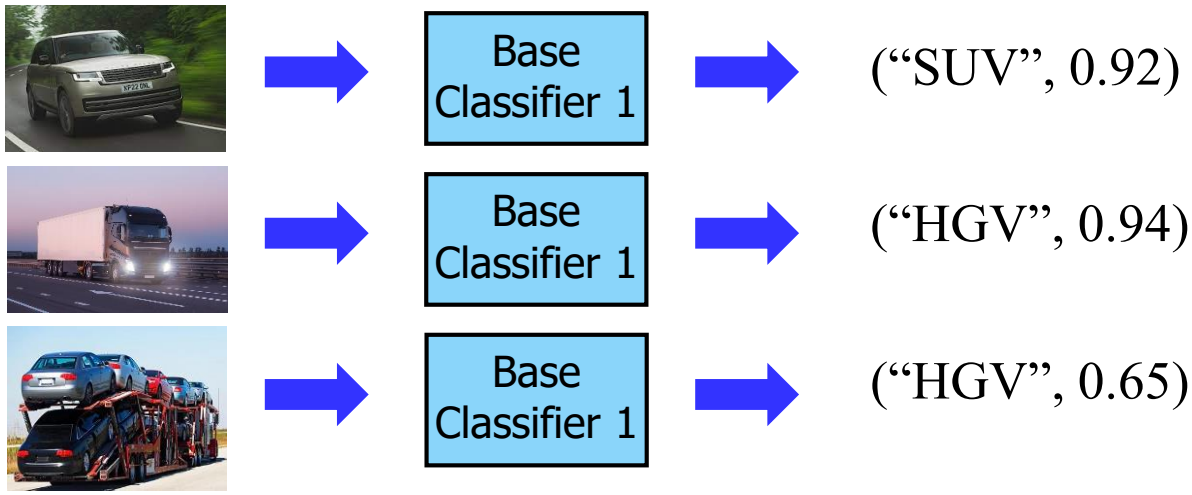
- **Focus of this Research**
  - Perception in autonomous mobile Cyber-Physical Systems is typically performed using classifiers that are based on Deep Learning (Deep Neural Networks)
  - How to minimize the **expected duration** (average time) to successful classification, subject to constraints on **classification quality**, and optionally a hard constraint on the **worst-case latency**

- **Approach**
  - Uses an ensemble of classifiers that can individually return either a class or say "I Don't Know" (**IDK**)
  - The problem is to decide on the optimal order in which the **IDK classifiers** should run to minimize the overall **expected duration** subject to constraints on **classification quality** and **worst-case latency**, given the classifier's **execution times**, **success probabilities**, and **dependences**

Background: Classifiers

- **Base Classifier**
  - Takes an **input sample** and outputs the **class** that it determines most likely matches the input sample, and a **confidence** that indicates how confident it is that the input sample belongs to the class

- **Examples**



Background: Classifiers

- **IDK Classifier**
  - Augments a base classifier with a **confidence threshold:** If the confidence is below the threshold then it outputs "I Don't Know" (**IDK**) otherwise it outputs the class
  - A **precision threshold** (not shown) indicates the long run proportion of a classifier's non-IDK outputs that must be correct, in other words in agreement with the ground truth, and is used to set the confidence threshold (as discussed later)
- **Examples**

Confidence threshold = 0.9



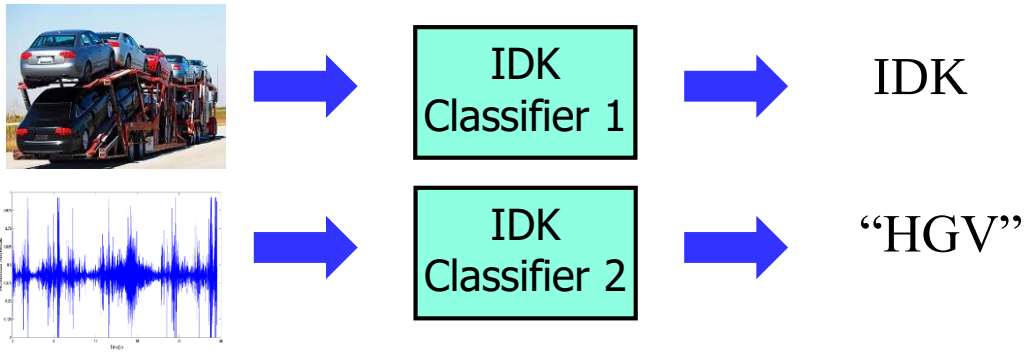
IDK Classifier 1 → "SUV"

IDK Classifier 1 → "HGV"

IDK Classifier 1 → IDK

Background: Classifiers

- **Variety of Classifiers**
  - Classifiers can make different trade offs between accuracy and execution time
  - For example by using more layers in a Deep Neural Network (DNN) to obtain more accurate results at the expense of a longer run time
  - Classifiers can also be used on different granularity images, for example 64 x 64, 256 x 256, or 1024 x 1024 pixels, to trade-off run-time versus accuracy
  - Classifiers can also use completely different types of data from an input sample, for example vision, acoustic, and seismic data
- **Examples**



IDK Classifier Model

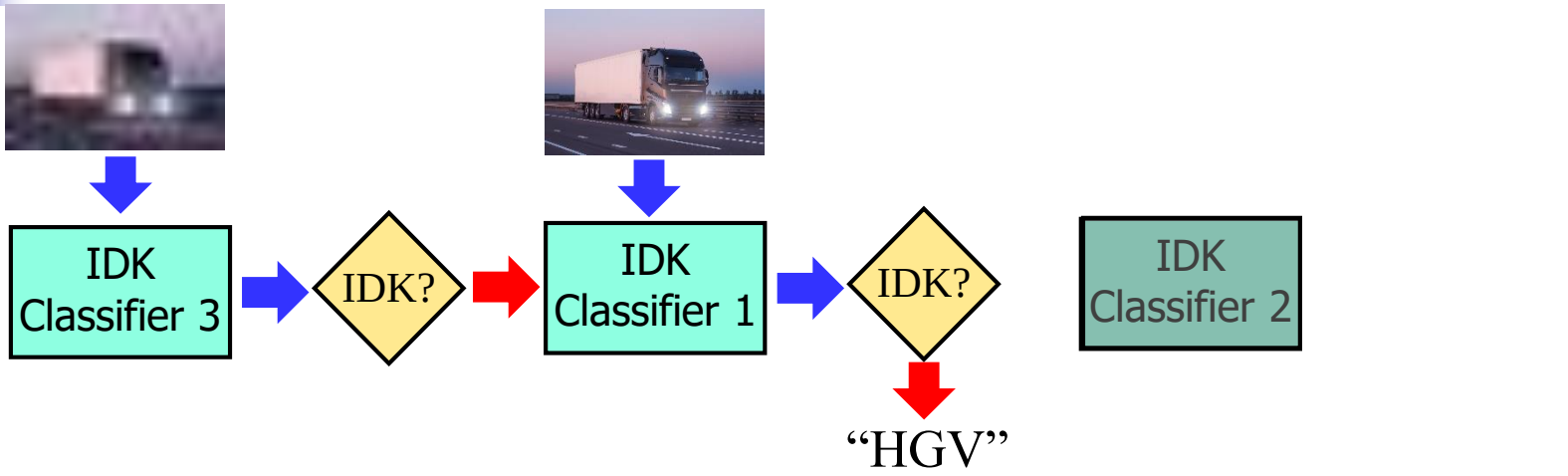
- **Probability of Successful Classification**
  - Each IDK classifier (K_i) is characterized by an average execution time $\bar{C}_i$, a worst-case execution time C_i and a probability P_i of successful classification (i.e. returning a class rather than IDK)
- **Dependences**
  - In some cases these probabilities may be pair-wise **independent,** meaning that the probability that one classifier will output a real class is independent of whether it is run on all inputs or on only those inputs where the other classifier outputs IDK
  - Classifiers that operate on disparate parts of an input sample, for example image and seismic data, may in some cases be independent
  - Classifiers that operate on different resolutions of the same image are typical not independent, but rather may be **fully dependent**, meaning that the classifier operating at a lower resolution is only able to successfully classify a strict subset of the input samples that the more powerful classifier operating at a higher resolution can identify
  - In general classifiers may exhibit **arbitrary dependences (**introduced by the environment, by the training process, and by common components and algorithms)

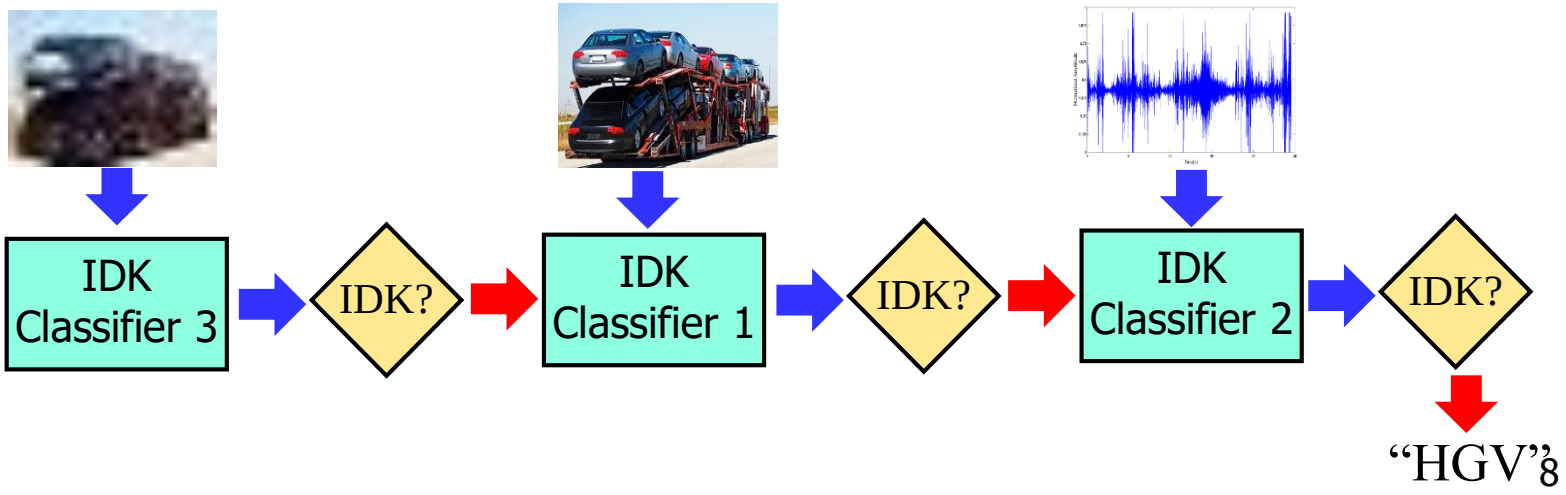
This work considers classifiers which exhibit **arbitrary dependences** between their probabilities of successful classification

Scheduling IDK Classifiers

- **IDK Cascades**
  - An IDK cascade is an ordered list of IDK classifiers
  - On a single processor, an IDK cascade comprises a linear sequence of IDK classifiers that is invoked as follows for any input sample that needs to be classified
    1. The first classifier in the IDK cascade is executed
    2. If it outputs a real class, rather than IDK, then the IDK cascade terminates and characterizes the input sample as being of the identified class
    3. Otherwise, the classifier outputs IDK, the subsequent classifier in the IDK cascade runs and the process continues from step 2.
  - If it is a requirement that all inputs are successfully classified, then the last classifier in the IDK cascade must always succeed
  - A classifier that always succeeds is referred to as a **deterministic classifier,** and has P_i=1
  - Alternatively, the overall probability of successful classification may be required to reach some **classification threshold** (L), such that the long run proportion of inputs to the IDK cascade that result in a real class is at least L

IDK Cascades: Examples





Validation: ResNet Case Study

- **ResNet**
  - Case study from the domain of (camera) image classification
  - Used classifiers based on the ResNet Deep Residual Network:
    - ResNet-18
    - ResNet-34
    - ResNet-50
    - ResNet-152
  - ResNet-*x* implies *x* layers in the neural network, with larger numbers of layers typically improving classification quality at the expense of a longer execution time
  - Used a representative data-set of 50,000 test images from the validation set of the ImageNet Large Scale Visual Recognition Challenge
  - Classifiers were run on an NVIDIA Jetson TX2

Validation: Multi-Modal Case Study

- **Multi-Modal**
  - Case study based on data from a project that seeks to autonomously detect enemy vehicles in a battlefield environment
  - Uses multi-modal classifiers that analyse camera images, acoustic, and seismic data
  - Initially four classifiers were studied (see paper for more details):
    - deepsense_both_contras,
    - cnn_acoustic,
    - deepsense_seismic,
    - yolov5s-compressed
  - Up to nine classifiers in all, using different combinations of modal data and compression
  - Made use of the entire available data-set of 1800 input samples
  - Classifiers were run on a Raspberry Pi 4

Overview of the Approach

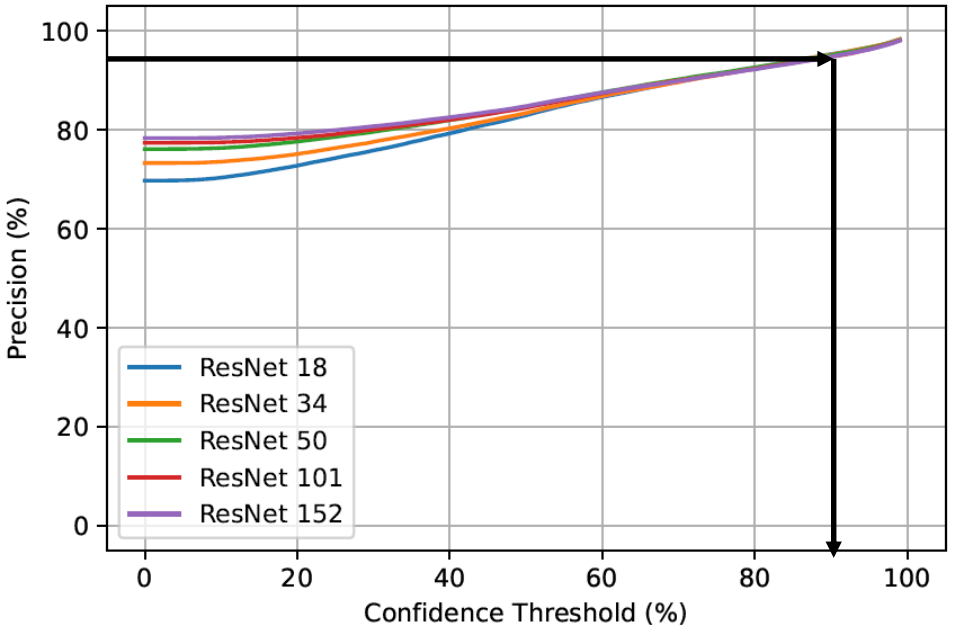
- **Profiling Phase**
  - Creates a **profile table** for each classifier based on representative input data augmented by ground truths
  - Uses the information in the profile table to set the **confidence threshold** for each IDK classifier based on a required **precision threshold**
  - Derives a **probability table** from the profile tables for all the IDK classifiers, capturing their dependences
- **Synthesis Phase**
  - Builds a DAG-based representation of all possible **IDK cascades**
  - Evaluates the cost of edges in the DAG using information from the **probability table**
  - Prunes the vertices and edges to cater for a **latency constraint** and a **classification threshold**
  - Uses a standard topological ordering algorithm to determine an **Optimal IDK Cascade** and its characteristics

Profiling Phase: Profile Table

- **Initialize the Profile Table**
  - The base classifiers are profiled against *N* **representative** input samples with known ground truths
  - Each input sample contains data for all modalities of the classifiers, for example image, acoustic, seismic and so on
  - The set of input samples must be representative of the population of all possible inputs (typically the data sets used in verifying classifier performance can be re-used for profiling)
  - This creates a **profile table** with a row for each of the *N* input samples giving the (class, confidence) pair output by each of the base classifiers along with the ground truth class
  - The **precision threshold** is then used to set the **confidence threshold** H_i for each IDK classifier, with input samples assigned a confidence below the confidence threshold resulting in IDK being returned
    A **precision threshold** of 0.95 ensures that when an IDK classifier returns a real class, rather than IDK, we can expect it to be correct (i.e. in accordance with the ground truth) 95% of the time

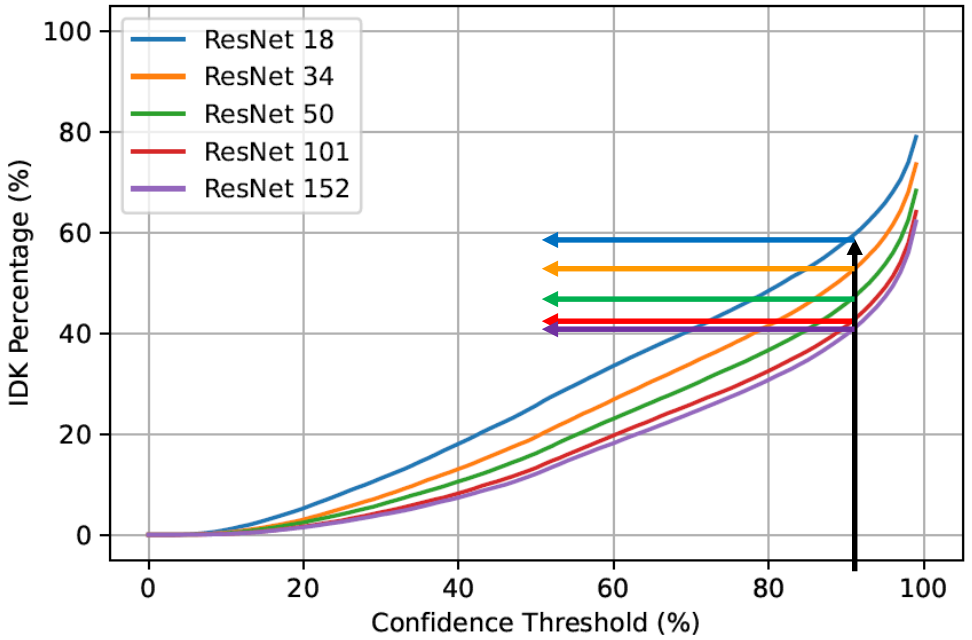
Profiling Phase: Confidence Thresholds

- **Set the Confidence Thresholds**
  - A minimum **confidence threshold** H_i is set for each classifier such that the proportion of input samples with a confidence above the threshold that are correctly classified (according to the ground truth) is no less than the required **precision threshold** (e.g. 0.95)
  - A **precision threshold** of 0.95 results in a **confidence threshold** of about 0.9 for all of the ResNet classifiers



Profiling Phase: Confidence Thresholds

- **Effect on the proportion of IDKs**
  - A **confidence threshold** of about 0.9 for the ResNet classifiers results in different proportions of IDKs returned
  - This is because the confidence levels of classifiers with fewer layers are typically lower resulting in more IDKs at the same threshold
  - For example, we obtain nearly 60% IDKs with ResNet-18, compared to about 40% IDKs with ResNet-152



Profiling Phase: Probability Table

- **Probability Table**
  - Once a **confidence threshold** has been set for each classifier, the **profile table** can be updated to indicate the inputs samples for which each classifier returns IDK
  - A **probability table** can then be constructed indicating:
    (i) the probability **Prob-S** that exactly the classifiers with a 1 in the *specified* pattern will return a real class and those with a 0 will return IDK
    (ii) the probability **Prob-A** that *any* one of the classifiers with a 1 in the pattern will return a real class

ResNet Case Study: Probability Table

ResNet -18 *A*	-34 *B*	-50 *C*	-152 *D*	Count	Prob-S	Prob-A
0	0	0	0	15880	0.3176	0
0	0	0	1	3011	0.06022	0.5902
0	0	1	0	1423	0.02846	0.545
0	0	1	1	2465	0.0493	0.64564
0	1	0	0	914	0.01828	0.49216
0	1	0	1	960	0.0192	0.63488
0	1	1	0	545	0.0109	0.60016
0	1	1	1	3382	0.06764	0.66942
1	0	0	0	649	0.01298	0.4284
1	0	0	1	452	0.00904	0.62476
1	0	1	0	304	0.00608	0.5847
1	0	1	1	1208	0.02416	0.66412
1	1	0	0	275	0.0055	0.54442
1	1	0	1	609	0.01218	0.65394
1	1	1	0	500	0.01	0.62218
1	1	1	1	17423	0.34846	0.6824
Totals				50000	1.00	

Classifier	*A*	*B*	*C*	*D*	*E*
H_i	0.890	0.895	0.896	0.910	1
$\bar{C}_i$ (ms)	16.9	27.8	37	101.1	1000
C_i (ms)	22.6	37.5	49.5	125.1	1000

Profiling Phase : Probability Table

- **Probability Table**
  - The Prob-S values come directly from the frequency of occurrence of the pattern in the **profile table**
  - The Prob-A values are computed from the Prob-S values in time that is quadratic in the number of rows, which is $O(4^n)$ overall, since there are 2^n rows in the table

Multi-Modal Case Study: Probability Table

A: deepsense_both_contras; B: cnn_acoustic;
C: deepsense_seismic; D: yolov5s-compressed

A	*B*	*C*	*D*	Count	Prob-S	Prob-A
0	0	0	0	56	0.031111111	0
0	0	0	1	33	0.018333333	0.298333333
0	0	1	0	35	0.019444444	0.736111111
0	0	1	1	18	0.01	0.816666667
0	1	0	0	11	0.006111111	0.221666667
0	1	0	1	5	0.002777778	0.461111111
0	1	1	0	5	0.002777778	0.807777778
0	1	1	1	4	0.002222222	0.868333333
1	0	0	0	181	0.100555556	0.907222222
1	0	0	1	76	0.042222222	0.940555556
1	0	1	0	698	0.387777778	0.941666667
1	0	1	1	304	0.168888889	0.962777778
1	1	0	0	82	0.045555556	0.921111111
1	1	0	1	31	0.017222222	0.949444444
1	1	1	0	195	0.108333333	0.950555556
1	1	1	1	66	0.036666667	0.968888889
TOTALS				1800	1.00	

Pattern

Probability Prob-S that classifiers B, C, and D return a real class and classifier A returns IDK

Probability Prob-A that at least one of B, C, or D returns a real class

Classifier	*A*	*B*	*C*	*D*	*E*
H_i	0.705	0.543	0.86	0.01	1
$\bar{C}_i$ (ms)	17.0	3.9	11.4	1440.8	5000
C_i (ms)	19.6	5.3	13.7	1613.2	5000

Profiling Phase: Dependences

- **Dependences in Behaviour**
  - Compared classifier behaviour (1: non-IDK, 0: IDK) for N input samples
  - Computed Pearson's correlation coefficient
  - ResNet classifiers shows strong (>0.5) degrees of correlation (red)
  - Multi-Modal classifiers show moderate (0.1-0.5) (orange), weak (0.05-0.1) (yellow), very weak (<0.05) (green) degrees of correlation

ResNet: Pearson's correlation coefficient

	A	B	C	D
A	1	0.668	0.630	0.579
B	0.668	1	0.678	0.639
C	0.630	0.678	1	0.686
D	0.579	0.639	0.686	1

Multi-Modal: Pearson's correlation coefficient

	A	B	C	D
A	1	0.055	0.265	-0.042
B	0.055	1	-0.071	-0.037
C	0.265	-0.071	1	-0.009
D	-0.042	-0.037	-0.009	1

As expected, many pairs of classifiers exhibit **arbitrary dependences**

Profiling Phase: Dependences

- **Dependences in Execution Times**
  - Compared classifier execution times (1: > median, 0: ≤ median)
  - Computed Pearson's correlation coefficient
  - All classifiers shows weak or very weak degrees of correlation
  - Using a Chi-squared test indicates no evidence against a null hypothesis of independence for some pairs of classifiers, but not for others
  - In conclusion the majority of the execution time is independent, with a small component (<7%) that is dependent
  - Validates that an assumption of independence for execution times is a reasonable approximation

ResNet: Pearson's correlation coefficient

	A	B	C	D
A	1	-0.003	-0.005	0.007
B	-0.003	1	0.040	0.039
C	-0.005	0.040	1	0.018
D	0.007	0.039	0.018	1

Multi-Modal: Pearson's correlation coefficient

	A	B	C	D
A	1	-0.013	0.024	0.013
B	-0.013	1	0.062	-0.044
C	0.024	0.062	1	-0.013
D	0.013	-0.044	-0.013	1

Our subsequent analysis assumes that **execution times are independent** but the classifier behaviour is not

Analysing IDK Cascades

- **Probabilities**
  - Consider an IDK cascade of classifiers in order: $\langle K'_1, K'_2, \ldots, K'_{n'} \rangle$
  - Let $\widehat{P}[S]$ be the probability that at least one of the classifiers in the subset *S* returns a real class and not IDK, and is given by the **Prob-A** values in the **probability table**
  - The expected duration of an IDK cascade is given by:

$$\sum_{i=1}^{n'} \left(\bar{C}'_i \times \left(1 - \widehat{P}[\{K'_1, K'_2, \ldots, K'_{i-1}\}] \right) \right)$$

    since each IDK classifier only executes if all of the previously finished classifiers returned IDK
- **Previous work**
  - Prior research published in "Optimally ordering IDK classifiers subject to deadlines" showed how to compute these values for IDK classifiers with **independent** or **fully dependent** behaviour and provided algorithms for determining optimal IDK cascades in those cases

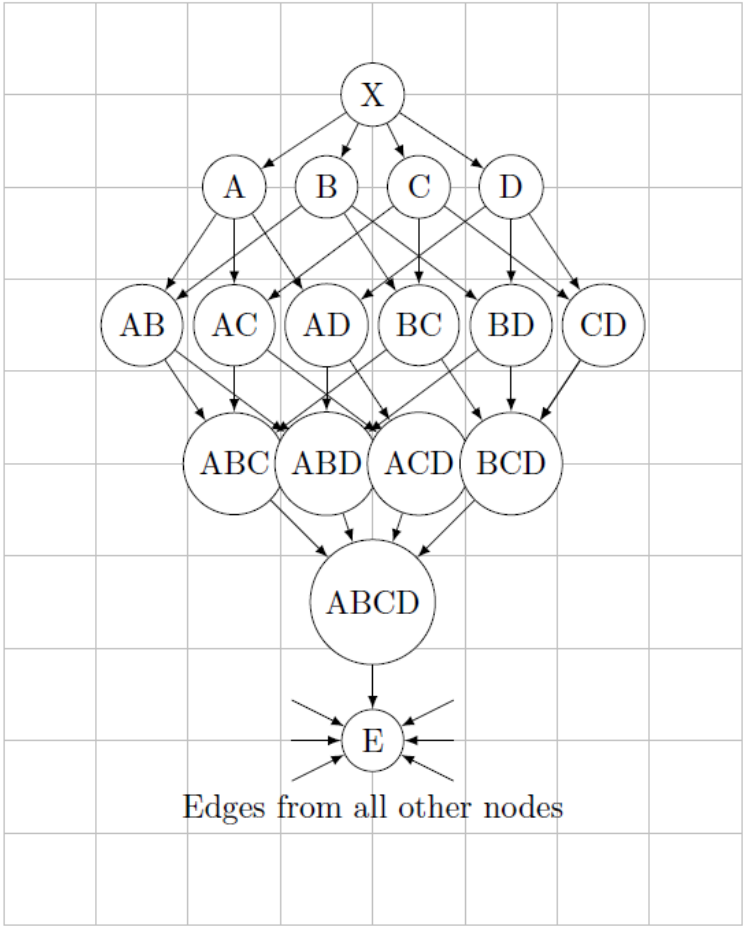
This research provides analysis and algorithms for determining optimal IDK cascades in the practical case of IDK classifiers with **arbitrary dependences**

Synthesizing Optimal IDK Cascades

- **Analysis**
  - For any arbitrary IDK cascade, we can compute:
    (i) the expected duration (formula on previous slide)
    (ii) the worst-case duration (sum of the execution times), and
    (iii) the probability of successful classification ($\widehat{P}[S]$, i.e. Prob-A, where *S* is the set of all classifiers in the cascade)
  - Hence we can determine if each IDK cascade complies with a given **latency constraint** on its worst-case duration, and a given **classification threshold** on the overall probability of successful classification
- **Permutations**
  - The number of possible IDK cascades grows as a factorial with increasing *n* (number of classifiers)
  - A more nuanced approach, than brute-force evaluation of all possibilities, is needed to find an **optimal IDK cascade**, i.e. with the minimum expected duration that complies with the constraints

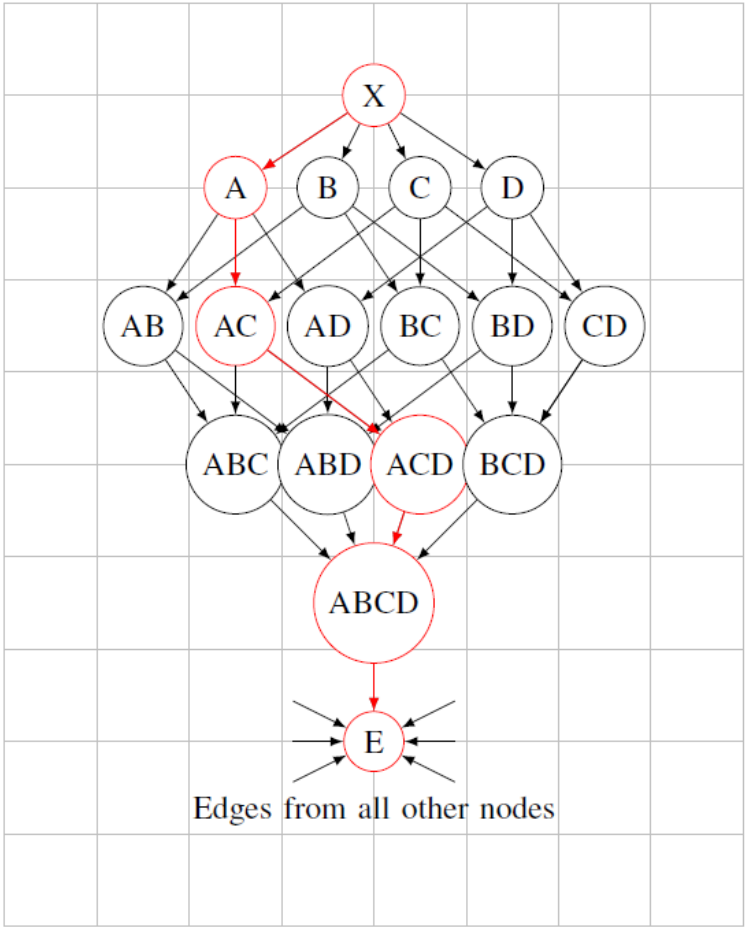
Synthesizing Optimal IDK Cascades

- **DAG-based representation**
  - To avoid factorial complexity we employ a graph-based representation in the form of a Directed Acyclic Graph (DAG)
  - Each vertex corresponds to a unique **subset** of the *n* IDK classifiers, hence there are $2^n - 1$ such vertices
  - We also include a start vertex X and an exit vertex E
  - A directed edge connects each vertex representing a subset of IDK classifiers with each of the vertices that represents the same subset extended via the addition of exactly one further classifier
  - In addition there is a directed edge from all other vertices to the exit vertex E



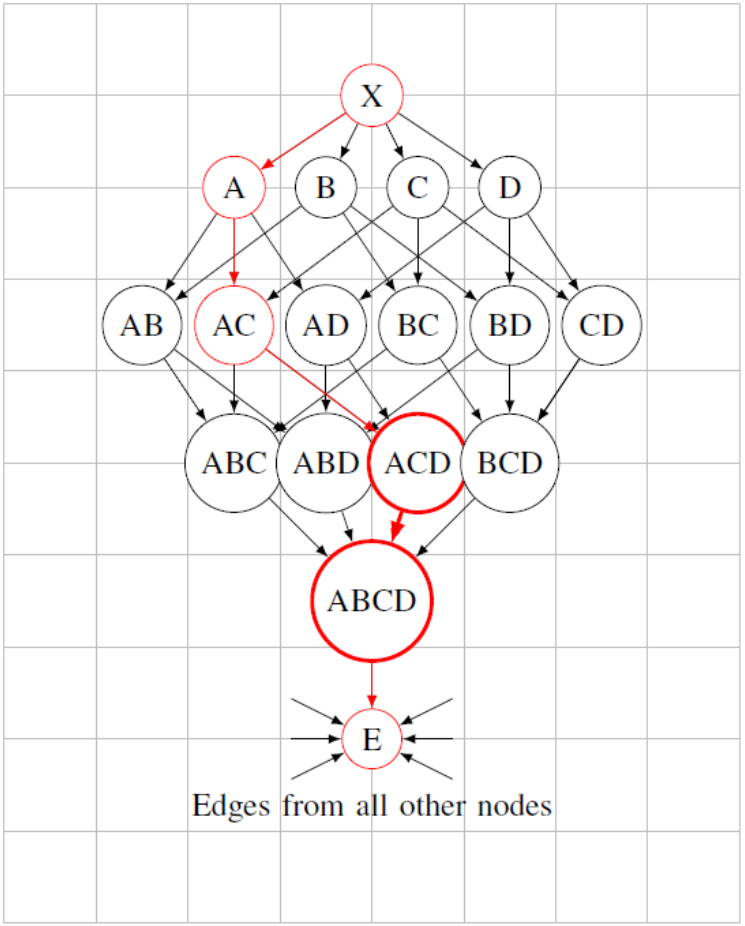
Synthesizing Optimal IDK Cascades

- **Paths and Edges**
  - Each unique permutation of classifiers forming an IDK cascade corresponds to a path through the DAG, from start to exit
  - On a given path the corresponding IDK cascade can be recovered by collecting the classifiers that are added in moving from one vertex to the next
  - For example the path X → A → AC → ACD → ABCD → E (highlighted in red) corresponds to the IDK cascade ⟨A,C,D,B,E⟩
  - A directed edge represents the cost of adding the new classifier to any sub-sequence formed from all of the classifiers in the subset *S* represented by the previous vertex



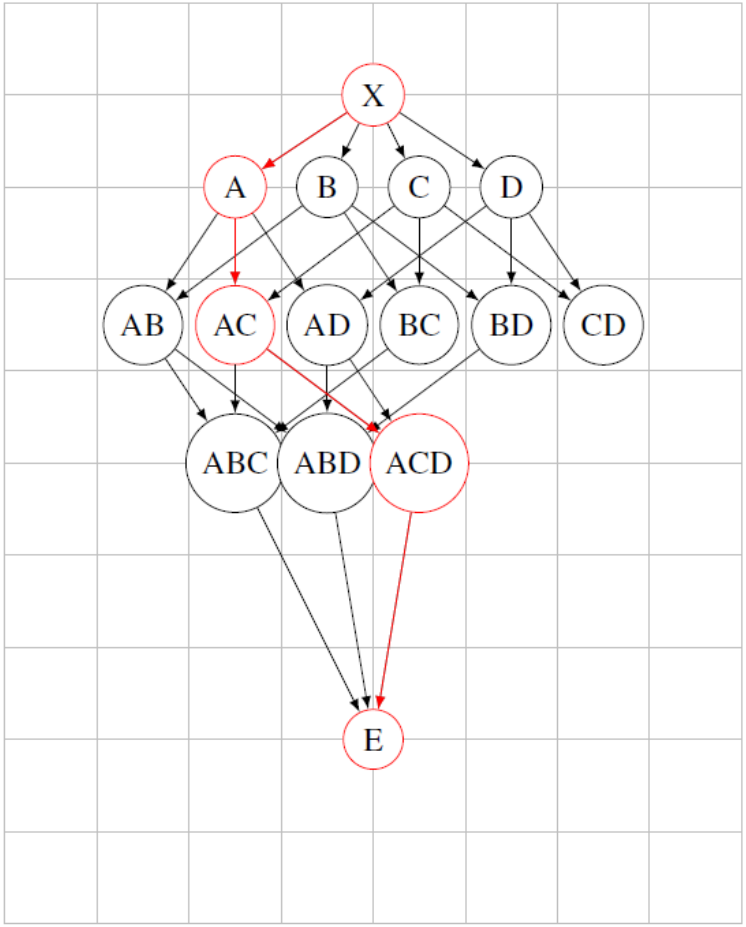
Synthesizing Optimal IDK Cascades

- **Paths and Edges**
  - For example, the edge from ACD to ABCD represents the addition of classifier B with a cost of $\bar{C}_B \times \left(1 - \hat{P}[\{A, C, D\}]\right)$
  - Importantly, this is the case irrespective of which path was taken to reach vertex ACD, since the cost depends only on the set of classifiers ACD, and not on their order
  - Once the DAG has been constructed, the problem of finding the **optimal IDK cascade** is reduced to finding the shortest path, which can be achieved, in time linear in the number of edges plus vertices, using standard topological ordering algorithms
  - Complexity is thus O($n2^n$), i.e. exponential rather than factorial, hence the method is effective for up to 20 classifiers



Synthesizing Optimal IDK Cascades

- **Latency Constraint**
  - Can be accounted for by summing the execution time of all classifiers in each vertex and then deleting those vertices (and adjoining edges) where this worst-case duration exceeds the latency constraint
  - For example in the DAG shown, BCD and ABCD have been removed
- **Classification Threshold**
  - Can be accounted for by determining the probability of success $\hat{P}[S]$ for the subset S of classifiers in each vertex, and then deleting the edge to the exit from each vertex where this probability is less than the classification threshold
  - For example, in the DAG shown only ABC, ABD, and ACD meet the classification threshold and have edges to the exit vertex

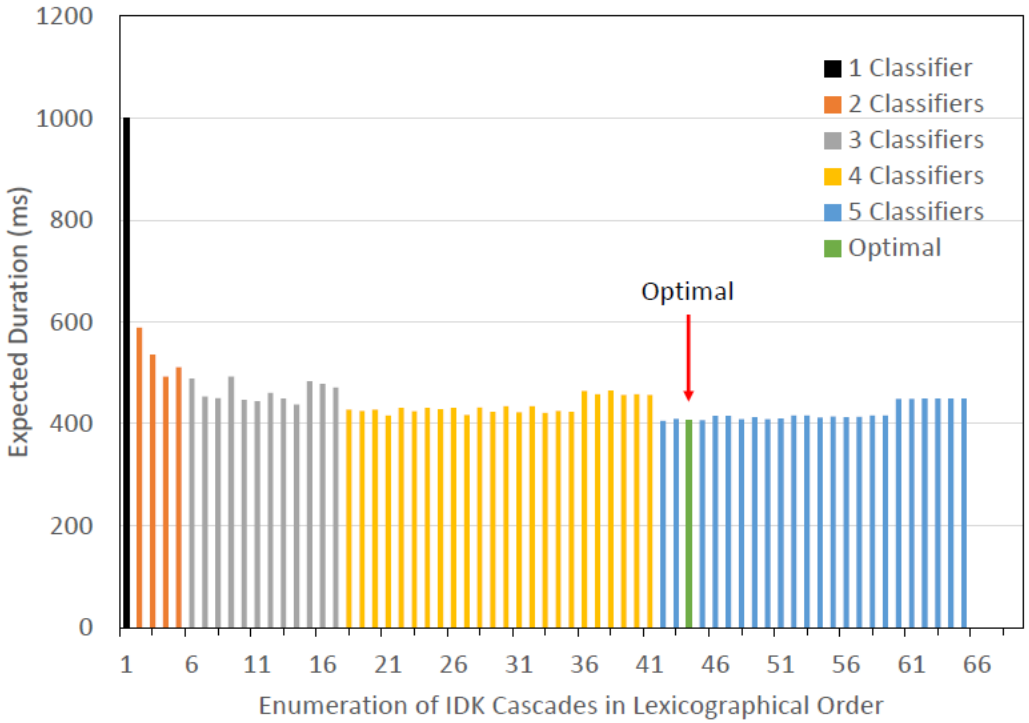


Case Study: ResNet

- **Classifiers**
  - (A) ResNet-18,
    (B) ResNet-34,
    (C) ResNet-50,
    (D) ResNet-152, and
    (E) a hypothetical deterministic classifier
  - 65 possible IDK cascades
- **Optimal IDK cascade**
  - ⟨A,C,B,D,E⟩ with an expected duration of 405.39ms
  - Assuming independence would result in ⟨A,B,C,D,E⟩ being wrongly selected as optimal, with an underestimate of 111ms for the expected duration

ResNet -18 *A*	-34 *B*	-50 *C*	-152 *D*	Count	Prob-S	Prob-A
0	0	0	0	15880	0.3176	0
0	0	0	1	3011	0.06022	0.5902
0	0	1	0	1423	0.02846	0.545
0	0	1	1	2465	0.0493	0.64564
0	1	0	0	914	0.01828	0.49216
0	1	0	1	960	0.0192	0.63488
0	1	1	0	545	0.0109	0.60016
0	1	1	1	3382	0.06764	0.66942
1	0	0	0	649	0.01298	0.4284
1	0	0	1	452	0.00904	0.62476
1	0	1	0	304	0.00608	0.5847
1	0	1	1	1208	0.02416	0.66412
1	1	0	0	275	0.0055	0.54442
1	1	0	1	609	0.01218	0.65394
1	1	1	0	500	0.01	0.62218
1	1	1	1	17423	0.34846	0.6824
Totals				50000	1.00	

Classifier	*A*	*B*	*C*	*D*	*E*
H_i	0.890	0.895	0.896	0.910	1
$\bar{C}_i$ (ms)	16.9	27.8	37	101.1	1000
C_i (ms)	22.6	37.5	49.5	125.1	1000



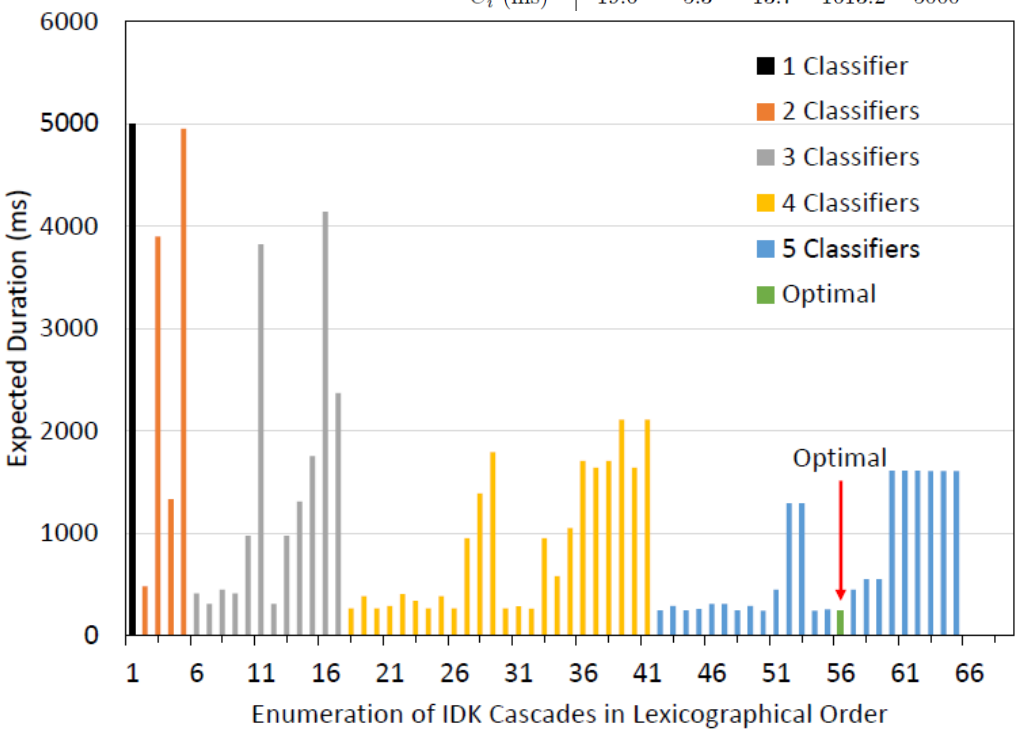
Case Study: Multi-Modal

- **Classifiers**
  - (A) deepsense_both_contras,
    (B) cnn_acoustic,
    (C) deepsense_seismic,
    (D) yolov5s-compressed, and
    (E) a hypothetical deterministic classifier
  - 65 possible IDK cascades
- **Optimal IDK cascade**
  - ⟨C,B,A,D,E⟩ with an expected duration of 242.5ms
  - Assuming independence would result in the same IDK cascade being selected as optimal, with an underestimate of 110.2ms for the expected duration

A: deepsense_both_contras; B: cnn_acoustic;
C: deepsense_seismic; D: yolov5s-compressed

A	*B*	*C*	*D*	Count	Prob-S	Prob-A
0	0	0	0	56	0.031111111	0
0	0	0	1	33	0.018333333	0.298333333
0	0	1	0	35	0.019444444	0.736111111
0	0	1	1	18	0.01	0.816666667
0	1	0	0	11	0.006111111	0.221666667
0	1	0	1	5	0.002777778	0.461111111
0	1	1	0	5	0.002777778	0.807777778
0	1	1	1	4	0.002222222	0.868333333
1	0	0	0	181	0.100555556	0.907222222
1	0	0	1	76	0.042222222	0.940555556
1	0	1	0	698	0.387777778	0.941666667
1	0	1	1	304	0.168888889	0.962777778
1	1	0	0	82	0.045555556	0.921111111
1	1	0	1	31	0.017222222	0.949444444
1	1	1	0	195	0.108333333	0.950555556
1	1	1	1	66	0.036666667	0.968888889
	Totals			1800	1.00	

Classifier	*A*	*B*	*C*	*D*	*E*
H_i	0.705	0.543	0.86	0.01	1
$\bar{C}_i$ (ms)	17.0	3.9	11.4	1440.8	5000
C_i (ms)	19.6	5.3	13.7	1613.2	5000

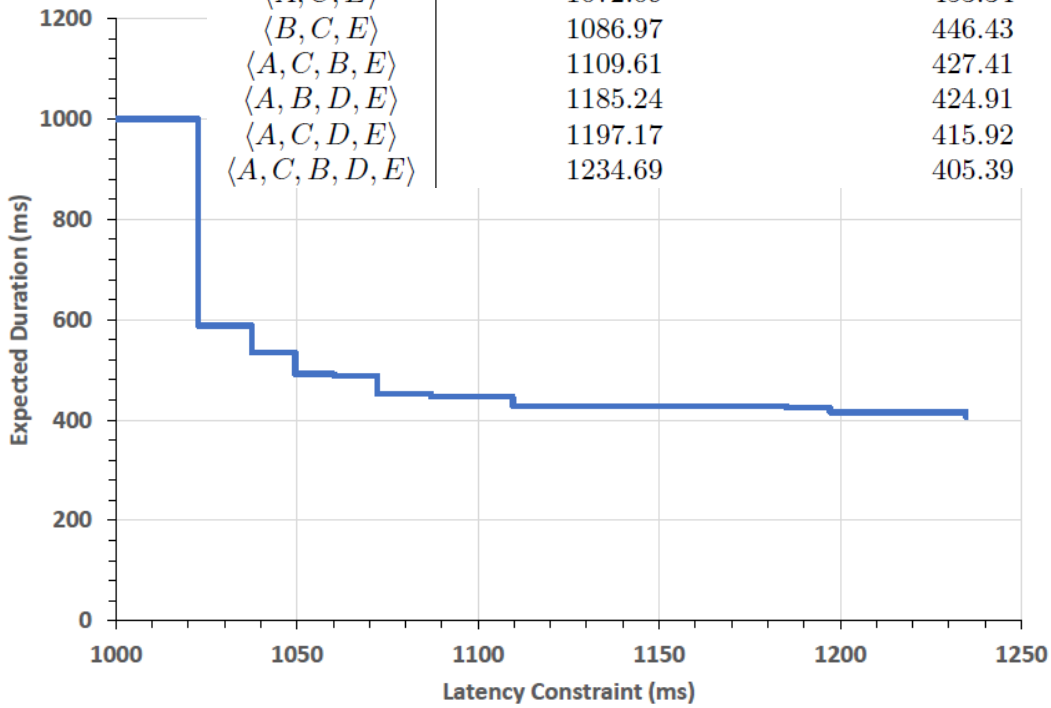


Case Study: ResNet

- **Latency Constraint**
  - Max latency of 1100ms
  - ⟨B,C,E⟩ is the optimal IDK cascade, with an expected duration of 446.43ms
- **Pareto Front**
  - Graph shows how the minimum expected duration (y-axis) varies with the latency constraint (x-axis)
  - The longer the permitted latency, the more the expected duration can be reduced
  - 11 Pareto optimal IDK cascades

ResNet: Pareto Optimal IDK Cascades

IDK Cascade	Worst-case (ms)	Expected Duration (ms)
$\langle E \rangle$	1000	1000
$\langle A, E \rangle$	1022.64	588.5
$\langle B, E \rangle$	1037.52	535.64
$\langle C, E \rangle$	1049.45	492
$\langle A, B, E \rangle$	1060.16	488.37
$\langle A, C, E \rangle$	1072.09	453.34
$\langle B, C, E \rangle$	1086.97	446.43
$\langle A, C, B, E \rangle$	1109.61	427.41
$\langle A, B, D, E \rangle$	1185.24	424.91
$\langle A, C, D, E \rangle$	1197.17	415.92
$\langle A, C, B, D, E \rangle$	1234.69	405.39

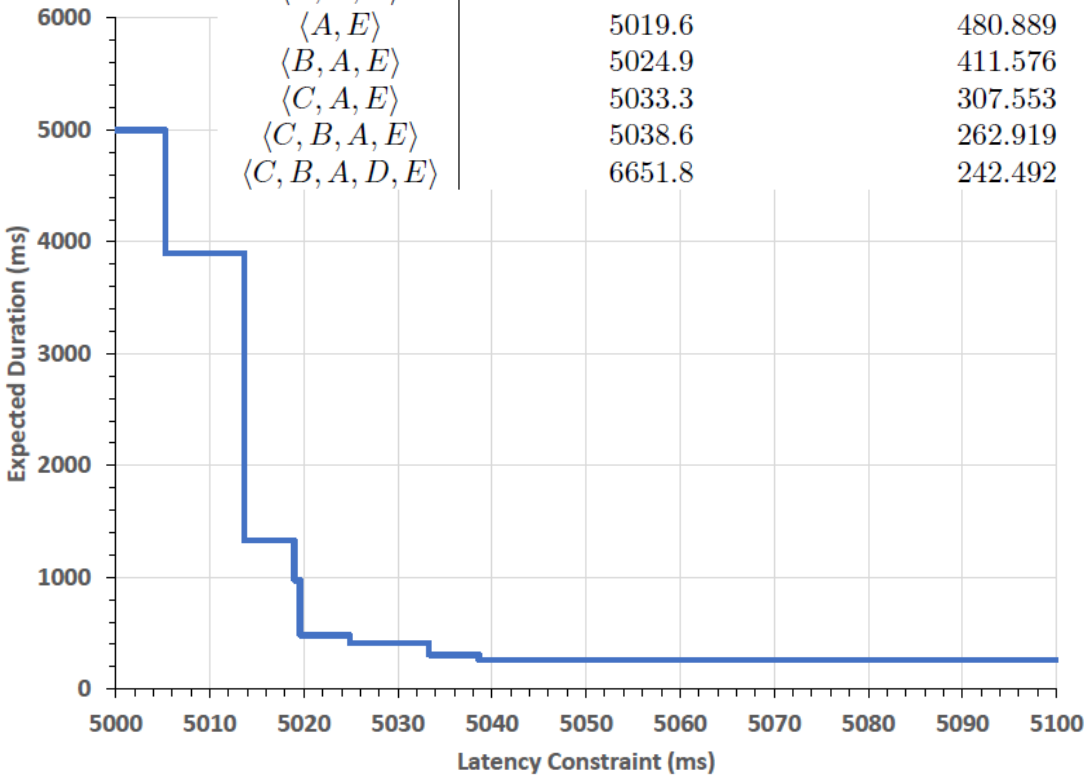


Case Study: Multi-Modal

- **Latency Constraint**
  - Max latency of 5030ms
  - ⟨B,A,E⟩ is the optimal IDK cascade, with an expected duration of 411.6ms
- **Pareto Front**
  - Graph shows how the minimum expected duration (y-axis) varies with the latency constraint (x-axis)
  - The longer the permitted latency, the more the expected duration can be reduced
  - 9 Pareto optimal IDK cascades

MultiModal: Pareto Optimal IDK Cascades

IDK Cascade	Worst-case (ms)	Expected Duration (ms)
$\langle E \rangle$	5000	5000
$\langle B, E \rangle$	5005.3	3895.567
$\langle C, E \rangle$	5013.7	1330.844
$\langle C, B, E \rangle$	5019	973.540
$\langle A, E \rangle$	5019.6	480.889
$\langle B, A, E \rangle$	5024.9	411.576
$\langle C, A, E \rangle$	5033.3	307.553
$\langle C, B, A, E \rangle$	5038.6	262.919
$\langle C, B, A, D, E \rangle$	6651.8	242.492



Validation: ResNet

- **Validation**
  - Used 10,000 images from the "TopImages" version of the ImageNetV2 data set

IDK Cascade	$\langle A, B, D \rangle$	$\langle A, C, D \rangle$	$\langle B, C, D \rangle$	$\langle A, B, C, D \rangle$
Expected Duration (ms)	788.50	800.36	870.14	878.45
Average Duration (ms)	766.32	782.44	850.41	853.70
Percentage Difference	2.81%	2.24%	2.27%	2.82%
Probability of Classification	0.65394	0.66412	0.66942	0.6824
Frequency of Classification	0.6266	0.6322	0.6379	0.6459
Difference	2.73%	3.19%	3.15%	3.65%

  - Difference in computed and actual duration and between computed and actual probability of classification is between 2% and 4%
  - Strong validation result given the disparate data sets used for profiling and validation

Case Study: Complete Multi-Modal

- **Nine IDK Classifiers**

Name	Index	Execution time (ms)	Confidence threshold	Success Probability
deepsense_both	A	17.5	0.66	0.899444
deepsense_both_contras	B	17.0	0.705	0.907222
deepsense_acoustic	C	11.7	0.715	0.213333
deepsense_seismic	D	11.4	0.86	0.736111
cnn_both	E	4.0	0.649	0.595556
cnn_acoustic	F	3.9	0.5433	0.220556
cnn_seismic	G	3.7	0.752	0.327222
yolov5s	H	3145.9	0.1	0.298889
yolov5s-compressed	I	1440.8	0.1	0.298333

- **Optimal IDK Cascades**

Classification Threshold	IDK cascade	Expected Duration (ms)	Worst-Case Duration (ms)	Probability of Classification
0.85	$\langle E, D \rangle$	8.61067	18.5	0.865556
0.9	$\langle E, D, F, G, C \rangle$	10.8212	42.8	0.9
0.925	$\langle E, B \rangle$	10.8756	24.4	0.932778
0.95	$\langle E, D, B \rangle$	10.8962	38.1	0.956667
0.975	$\langle E, D, A, B \rangle$	11.6546	59.5	0.981111
1	$\langle E, D, B, A, G, F, C, X \rangle$	89.7576	5083.8	1

Case Study: Complete Multi-Modal

- **Dependences: Behaviour**

	A	B	C	D	E	F	G	H	I
A	1	0.377	0.111	0.282	0.177	0.080	0.143	0.052	0.048
B	0.377	1	0.059	0.265	0.209	0.055	0.121	-0.043	-0.043
C	0.111	0.059	1	-0.067	0.219	0.701	-0.103	-0.028	-0.030
D	0.282	0.265	-0.067	1	0.127	-0.071	0.219	-0.005	-0.008
E	0.177	0.209	0.219	0.127	1	0.231	0.326	0.035	0.037
F	0.080	0.055	0.701	-0.071	0.231	1	-0.066	-0.036	-0.035
G	0.143	0.121	-0.103	0.219	0.326	-0.066	1	0.155	0.151
H	0.052	-0.043	-0.028	-0.005	0.035	-0.036	0.155	1	0.988
I	0.048	-0.043	-0.030	-0.008	0.037	-0.035	0.151	0.988	1

- **Dependences: Execution Times**

	A	B	C	D	E	F	G	H	I
A	1	0.031	0.036	-0.011	-0.027	-0.009	0.022	0.007	0.004
B	0.031	1	0.009	0.024	-0.040	-0.013	-0.024	-0.011	0.013
C	0.036	0.009	1	0.000	0.029	-0.004	0.031	-0.002	0.049
D	-0.011	0.024	0.000	1	-0.020	0.062	-0.058	0.020	-0.013
E	-0.027	-0.040	0.029	-0.020	1	-0.007	0.076	-0.018	0.007
F	-0.009	-0.013	-0.004	0.062	-0.007	1	0.024	-0.009	-0.044
G	0.022	-0.024	0.031	-0.058	0.076	0.024	1	-0.011	0.024
H	0.007	-0.011	-0.002	0.020	-0.018	-0.009	-0.011	1	-0.013
I	0.004	0.013	0.049	-0.013	0.007	-0.044	0.024	-0.013	1

IDK cascades on Multiple Processors

- **Model**
  - *m* processors, so up to *m* IDK classifiers can run in parallel (non-pre-emptively)
  - Each classifier (K_i) takes the same execution time $\bar{C}_i$ and has the same probability of successful classification (P_i), irrespective of which processor it executes on
  - Make a simplifying assumption of constant execution times $\bar{C}_i$ (or at least constant processor hold times)
- **Analysis**
  - The ordered finish times (f_i) for each classifier determine the **expected duration**, i.e. elapsed time required for the IDK cascade (irrespective of the scheduling policy or processor allocation):

$$f'_1 + \sum_{i=1}^{n-1} (f'_{i+1} - f'_i)(1 - \widehat{P}[\{K'_1, \ldots, K'_i\}]$$

  - The formula is derived by observing that the probability of execution continuing between one finish time f_i and the next f_{i+1} depends only on the probability of successful classification by those classifiers that have finished by the earlier finish time f_i
  - To cater for a classification threshold *L*, we can terminate the summation once the threshold has been reached i.e. once $\widehat{P}[\{K'_1, \ldots, K'_{i+1}\}] \geq L$

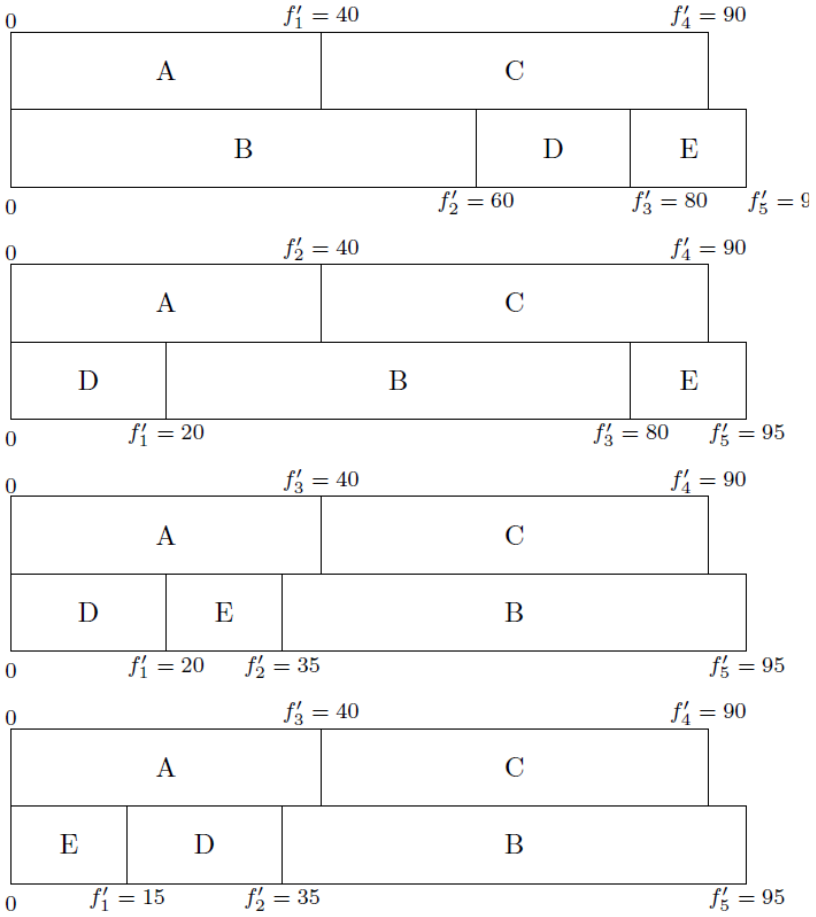
IDK cascades on Multiple Processors

By an IDK cascade on multiple processors, we mean an allocation of classifiers to processors and a schedule of classifiers on each processor

- **Theorems and Proofs (in the paper)**
  - **Lemma 1**: *An optimal IDK cascade is locally work conserving, in other words no processor becomes idle until all classifiers allocated to it have finished*
  - **Lemma 2:** *An optimal IDK cascade is globally work conserving, in other words it leaves no processor idle when there is a classifier to run*
  - **Theorem 1:** *List scheduling of an appropriate ordered list of classifiers suffices to provide an optimal IDK cascade*
    (List scheduling means that whenever a processor becomes idle, it takes the next available classifier in the ordered list and runs it)
- **Permutations**
  - Theorem 1 implies that an optimal IDK cascade can be found by considering each permutation of the *n* classifiers as a list and determining the corresponding allocation, schedule, and hence expected duration (from the finish time formula)
  - Issue is that this brute-force approach has factorial complexity, a more nuanced approach is therefore needed

Synthesizing an Optimal IDK Cascade for Multiple Processors

- **DAG-based representation**
- **Difficulties**
  - Unlike in the single processor case, classifiers do not necessarily finish in the same order as they start executing
  - For example the schedules on two processors for IDK cascades ⟨A,B,C,D,E⟩, ⟨A,D,B,C,E⟩, ⟨A,D,E,B,C⟩, and ⟨A,E,D,B,C,⟩ are shown in the diagram
  - In the first schedule, classifier C starts before D, but finishes later
  - This complicates the DAG-based representation as it needs to keep track of which classifiers are running and which have completed

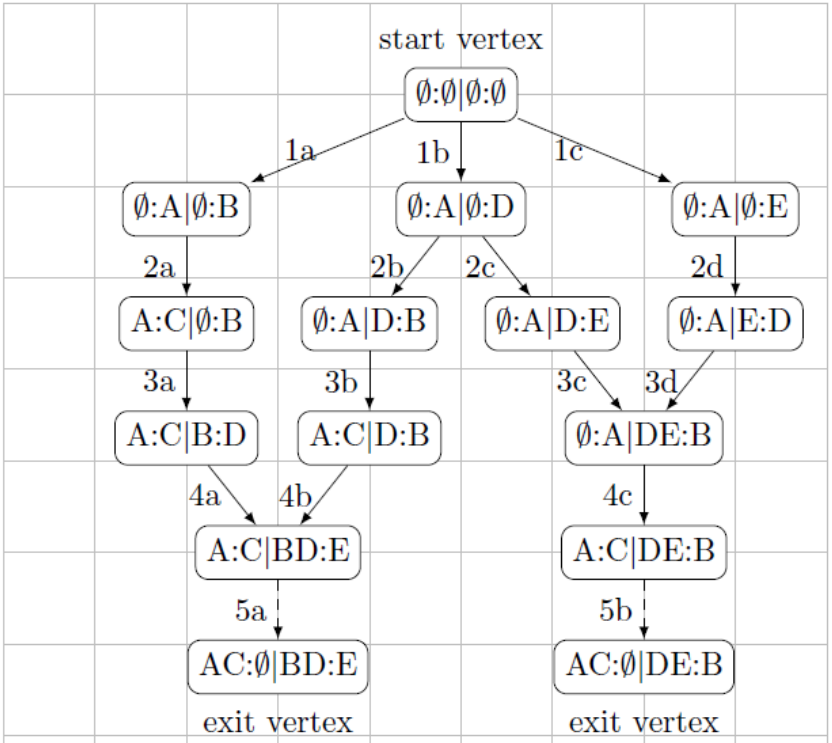


Synthesizing an Optimal IDK Cascade for Multiple Processors

- **DAG-based representation**
  - Each vertex represents *m* completed sets and *m* running sets (one per processor)
  - Each of the *n* classifiers may appear in at most one of these sets, with at most one classifier in each running set
  - Each vertex records a finishing time equal to the minimum *makespan* given the allocated classifiers in the sets
  - Also records the set *S* of all classifiers in the *m* completed sets and hence the probability of successful classificatio$\widehat{P}[S]$
  - Cost of an edge given by:

$$(f'_{i+1} - f'_i) \times \left(1 - \widehat{P}[S]\right)$$

based on the finish times at the previous and next vertices and the set *S* of classifiers at the previous vertex



Full details of the DAG-based representation and solution are given in the paper

Case Study: Complete Multi-Modal

- **Classifiers**

Name	Index	Execution time (ms)	Confidence threshold	Success Probability
deepsense_both	A	17.5	0.66	0.899444
deepsense_both_contras	B	17.0	0.705	0.907222
deepsense_acoustic	C	11.7	0.715	0.213333
deepsense_seismic	D	11.4	0.86	0.736111
cnn_both	E	4.0	0.649	0.595556
cnn_acoustic	F	3.9	0.5433	0.220556
cnn_seismic	G	3.7	0.752	0.327222
yolov5s	H	3145.9	0.1	0.298889
yolov5s-compressed	I	1440.8	0.1	0.298333

- **Classification Thresholds (single processor)**

Classification Threshold	IDK cascade	Expected Duration (ms)	Worst-Case Duration (ms)	Probability of Classification
0.85	$\langle E, D \rangle$	8.61067	15.4	0.865556
0.9	$\langle E, D, F, G, C \rangle$	10.8212	34.7	0.9
0.925	$\langle E, B \rangle$	10.8756	21	0.932778
0.95	$\langle E, D, B \rangle$	10.8962	32.4	0.956667
0.975	$\langle E, D, B, A \rangle$	11.6546	49.9	0.981111
1	$\langle E, D, B, A, G, F, C, X \rangle$	89.7576	5069.2	1

Case Study: Complete Multi-Modal

- **Classification Thresholds (multiple processors)**

Single processor

10.8962	32.4

100%
100%

Two processor

Classification Threshold	IDK Cascade	Processor 1	Processor 2	Expected Duration (ms)	Worst-Case Duration (ms)	Probability of Classification
0.85	$\langle E, G, D \rangle$	$\langle E, G \rangle$	$\langle D \rangle$	6.77911	11.4	0.876667
0.9	$\langle E, G, D, F, C \rangle$	$\langle E, G, C \rangle$	$\langle D, F \rangle$	7.69972	19.4	0.9
0.925	$\langle E, D, F, G, B \rangle$	$\langle E, B \rangle$	$\langle D, F, G \rangle$	8.16333	21	0.963889
0.95	$\langle E, D, F, G, B \rangle$	$\langle E, B \rangle$	$\langle D, F, G \rangle$	8.16333	21	0.963889
0.975	$\langle E, G, D, F, B, A \rangle$	$\langle E, G, B \rangle$	$\langle D, F, A \rangle$	8.5605	32.8	0.983889
1	$\langle E, G, D, F, B, A, C, I, H, X \rangle$	$\langle E, G, B, X \rangle$	$\langle D, F, A, C, I, H \rangle$	70.4877	5024.7	1

75%
65%

Three processor

Classification Threshold	IDK Cascade	Processor 1	Processor 2	Processor 3	Expected Duration (ms)	Worst-Case Duration (ms)	Probability of Classification
0.85	$\langle G, E, F, D \rangle$	$\langle G, F \rangle$	$\langle E \rangle$	$\langle D \rangle$	6.33206	11.4	0.892778
0.9	$\langle G, E, F, D, C \rangle$	$\langle G, C \rangle$	$\langle E, F \rangle$	$\langle D \rangle$	6.77161	15.4	0.9
0.925	$\langle G, E, F, D, C, B \rangle$	$\langle G, B \rangle$	$\langle E, F, C \rangle$	$\langle D \rangle$	7.33195	20.7	0.965556
0.95	$\langle G, E, F, D, C, B \rangle$	$\langle G, B \rangle$	$\langle E, F, C \rangle$	$\langle D \rangle$	7.33195	20.7	0.965556
0.975	$\langle G, E, F, D, B, C, A \rangle$	$\langle G, B \rangle$	$\langle E, F, A \rangle$	$\langle D, C \rangle$	7.50578	25.4	0.984444
1	$\langle G, E, F, D, B, A, C, I, H, X \rangle$	$\langle G, B, I \rangle$	$\langle E, F, A, C, H \rangle$	$\langle D, X \rangle$	69.2984	5011.4	1

67%
64%

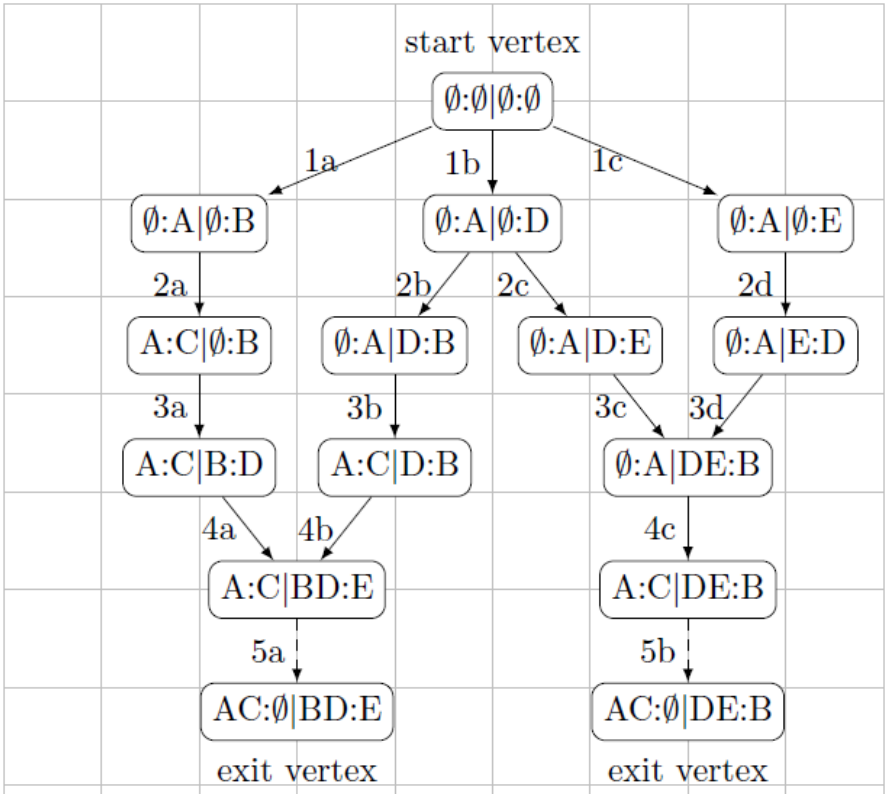
Four processor

Classification Threshold	IDK Cascade	Proc. 1	Proc. 2	Proc. 3	Proc. 4	Expected Duration (ms)	Worst-Case Duration (ms)	Probability of Classification
0.85	$\langle G, F, E, D \rangle$	$\langle G \rangle$	$\langle F \rangle$	$\langle E \rangle$	$\langle D \rangle$	6.18794	11.4	0.892778
0.9	$\langle G, E, F, D, C \rangle$	$\langle G, F \rangle$	$\langle E \rangle$	$\langle D \rangle$	$\langle C \rangle$	6.36422	11.7	0.9
0.925	$\langle G, E, F, D, C, B \rangle$	$\langle G, F \rangle$	$\langle E, C \rangle$	$\langle D \rangle$	$\langle B \rangle$	6.92311	17	0.965556
0.95	$\langle G, E, F, D, C, B \rangle$	$\langle G, F \rangle$	$\langle E, C \rangle$	$\langle D \rangle$	$\langle B \rangle$	6.92311	17	0.965556
0.975	$\langle G, E, F, D, B, C, A \rangle$	$\langle G, F, C \rangle$	$\langle E, A \rangle$	$\langle D \rangle$	$\langle B \rangle$	7.09133	21.5	0.984444
1	$\langle G, E, F, D, B, A, C, I, H, X \rangle$	$\langle G, F, X \rangle$	$\langle E, A, H \rangle$	$\langle D, I \rangle$	$\langle B, C \rangle$	68.8584	5007.6	1

64%
53%

Proof-of-Concept Implementation: DAG-based approach

- **Optimized C++ Implementation**
  - Computed the minimum cost for each vertex on-the-fly during construction of the DAG, with a pointer back to the previous vertex corresponding to that minimum, which avoids storing edges
  - Used a large hash table (2^{26} entries) to detect and eliminate duplicate vertices
  - Pruned vertices, during construction, to obtain compliance with the latency constraint and classification threshold
  - Used a simplified algorithm for the single processor case with a smaller hash table (2^{20} entries)
  - Run on a laptop PC (Lenovo ThinkPad, Intel Core i5 CPU 1.60GHz, 16 GBytes RAM, Windows 10)

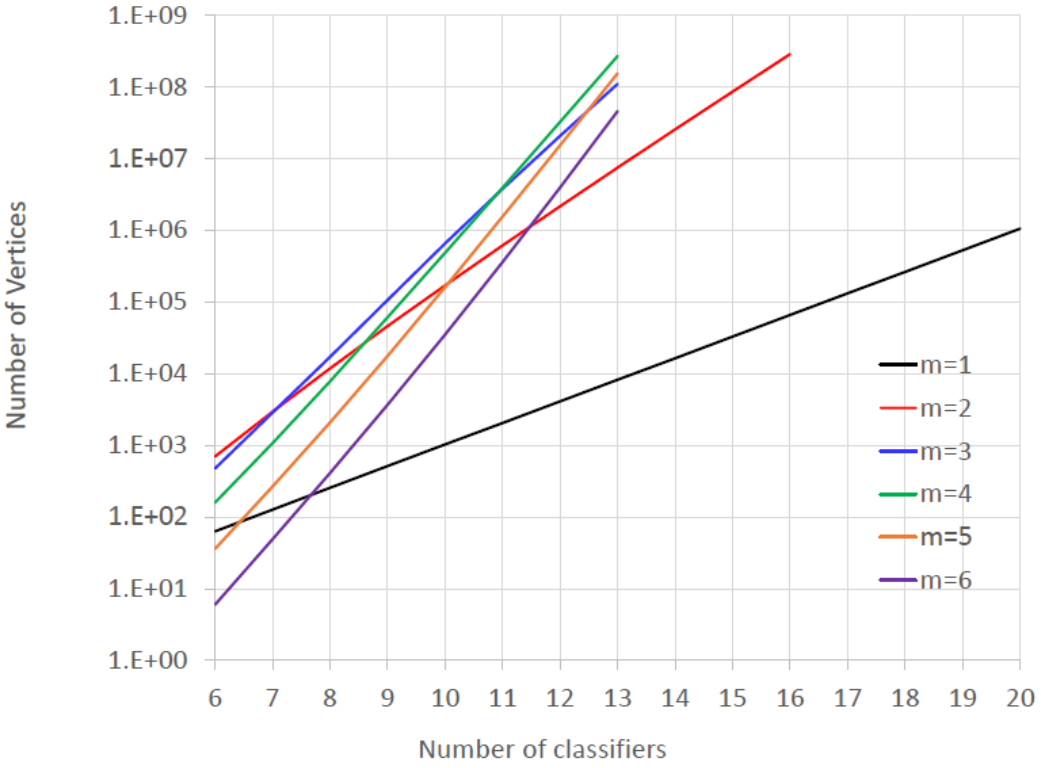


Full details of the DAG-based implementation are given in the paper

Proof-of-Concept Implementation: Efficiency

- **Synthetic test case**
  - Approximates the worst-case
  - Scalable to *n* classifiers
  - Assumes:
    The classifiers have disjoint probabilities of 1/*n* and a classification threshold of 1, so that all *n* classifiers have to be used
    No latency constraint, so no pruning of vertices
  - Limited memory usage to 24Gbytes (paged)
  - Graph shows that the space complexity is exponential, with straight lines against a log scale for the y-axis

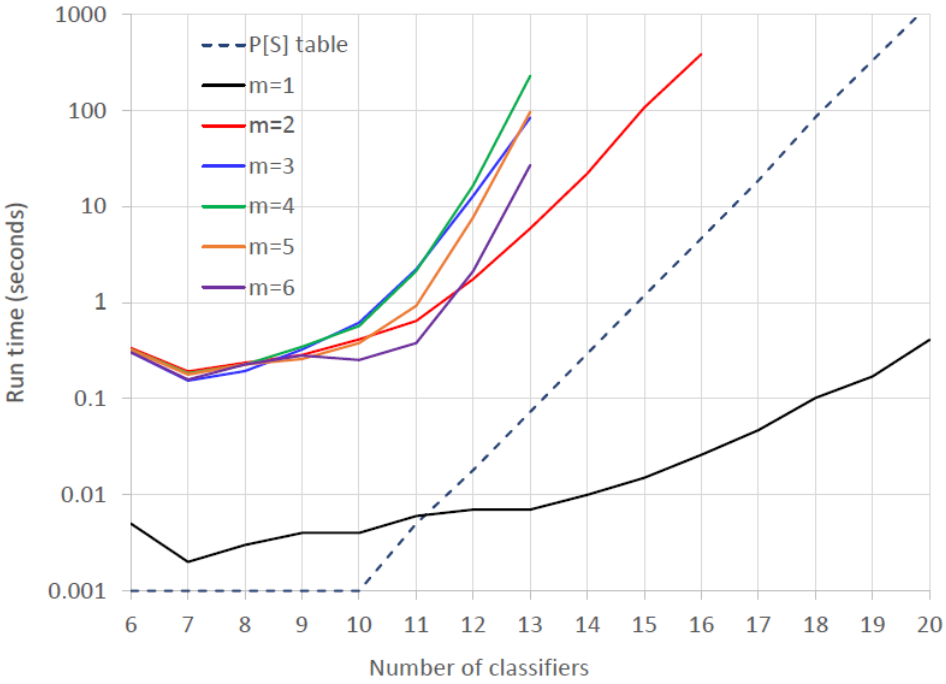
Space Complexity: Number of Vertices



Proof-of-Concept Implementation: DAG-based approach

- **Synthetic test case**
  - Limited run-time to about 1200 seconds (20 mins)
  - Decrease in run-time from 6 to 7 classifiers due to cache warm up
  - Roughly constant runtime for 6-11 classifiers due to initializing large hash table (constant overhead)
- **Summary**
  - Implementation supports:
    20 classifiers on 1 processors
    16 classifiers on 2 processors
    13 classifiers on 3-6 processors
  - Unlikely that more than about 12 classifiers would be used for the same problem – so this approach represents a practical solution

Time Complexity: Runtime

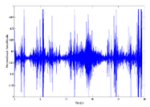


Conclusions

- **Summary**
  - Prior work showed how optimal IDK cascades can be synthesized for single processors, but made the unrealistic assumptions about independence or full dependence
  - This work enables the synthesis of optimal IDK cascades in the practical case of classifiers with arbitrary dependences, as well as introducing solutions for the case of multiple processors
  - It also caters for latency constraints and realistic classification thresholds, rather than requiring the presence of a deterministic classifier that can classify all inputs
  - The complexity of the approach is exponential (rather than factorial) and provides solutions for up to 20 classifiers on a single processor, and 13 or more classifiers on multiple processors
  - The problem is NP-complete on multiple processors (as shown in the paper), effectively necessitating exponential complexity to find optimal solutions
  - Effectiveness of the approach has been demonstrated on two real-world case studies: ResNet and Multi-Modal

	A	B	C	D	E	F	G	H	I
A	1	0.377	0.111	0.282	0.177	0.080	0.143	0.052	0.048
B	0.377	1	0.059	0.265	0.209	0.055	0.121	-0.043	-0.043
C	0.111	0.059	1	-0.067	0.219	0.701	-0.103	-0.028	-0.030
D	0.282	0.265	-0.067	1	0.127	-0.071	0.219	-0.005	-0.008
E	0.177	0.209	0.219	0.127	1	0.231	0.326	0.035	0.037
F	0.080	0.055	0.701	-0.071	0.231	1	-0.066	-0.036	-0.035
G	0.143	0.121	-0.103	0.219	0.326	-0.066	1	0.155	0.151
H	0.052	-0.043	-0.028	-0.005	0.035	-0.036	0.155	1	0.988
I	0.048	-0.043	-0.030	-0.008	0.037	-0.035	0.151	0.988	1

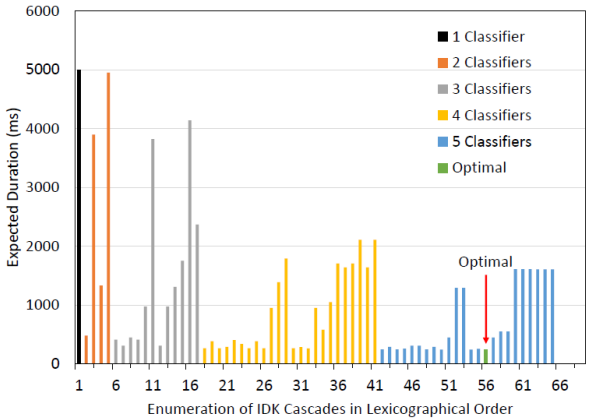
Discussion and Questions?

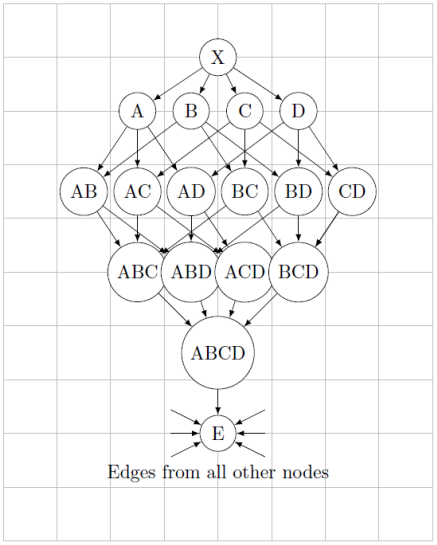


IDK Classifier 1 → IDK

IDK Classifier 2 → "HGV"

rob.davis@york.ac.uk





Classification Threshold	IDK cascade	Expected Duration (ms)	Worst-Case Duration (ms)	Probability of Classification
0.85	$\langle E, D \rangle$	8.61067	18.5	0.865556
0.9	$\langle E, D, F, G, C \rangle$	10.8212	42.8	0.9
0.925	$\langle E, B \rangle$	10.8756	24.4	0.932778
0.95	$\langle E, D, B \rangle$	10.8962	38.1	0.956667
0.975	$\langle E, D, A, B \rangle$	11.6546	59.5	0.981111
1	$\langle E, D, B, A, G, F, C, X \rangle$	89.7576	5083.8	1

	A	B	C	D
A	1	-0.003	-0.005	0.007
B	-0.003	1	0.040	0.039
C	-0.005	0.040	1	0.018
D	0.007	0.039	0.018	1