Research Prediction Competition

Google Landmark Retrieval Challenge

Given an image, can you find all of the same landmarks in a dataset?

$2,500
Prize Money

Google · 218 teams · 18 days ago

anokas
1st place

1st Place Solution Summary: CVSSP & Visual Atoms (0.627)

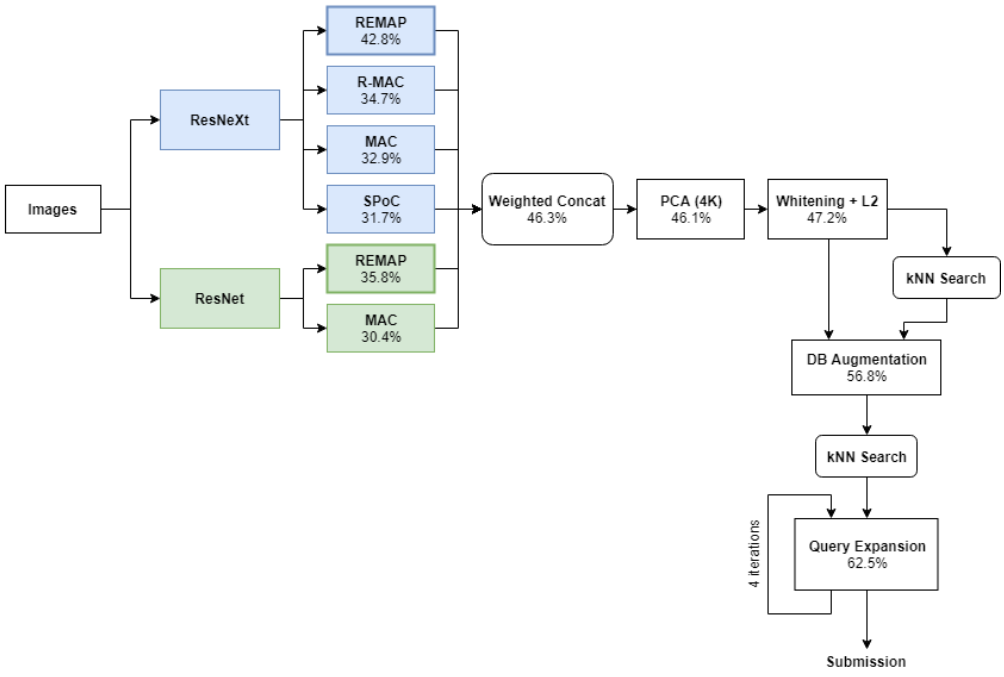
151

posted in Google Landmark Retrieval Challenge 18 days ago

Hi everyone,

Congratulations to everyone else who participated for making it such a fierce competition :)

Here's a detailed summary of our team's solution to the competition for those of you who are curious. Our solution consisted of two main components: First, creating a high performance global descriptor that can represent the images in the dataset as singular vectors, and then building an efficient scheme for matching these vectors to find the most likely matches to submit to the leaderboard.
Below is a high level flowchart overview, with LB scores labelled at each applicable step, followed by a detailed description of each component of our solution.



NB: Throughout this post we refer to scores as percentages for clarity, i.e. 62.5% = 0.625. All scores are on the public LB.

In-depth

Global Descriptors

The main part of our solution involves several global descriptors - vectors which describe the entire contents of the image. We start off with two pre-trained CNN models (ResNet and ResNeXt), and use four state-of-the-art aggregation methods to generate global descriptors from these models: Here are brief details of each method and its "raw" performance (i.e. without query expansion, database expansion, etc):

- **Region-Entropy based Multi-layer Abstraction Pooling (REMAP) [42.8% mAP]:** Our latest design for a global descriptor, which aggregates a hierarchy of deep features from different CNN layers, trained to represent multiple and complementary levels of visual abstraction. We plan to present the details of our REMAP architecture at the forthcoming CVPR workshop.
- **Maximum Activations of Convolutions (MAC) [32.9% mAP]:** MAC descriptors encode the maximum local response of each of the final layer convolutional filters. In the MAC architecture, the last convolutional layer of ResNeXt is followed by a max-pooling layer, L2-normalization layer and PCA+Whitening layer.
- **Sum-pooling of Convolutions (SPoC) [31.7% mAP]:** In the SPoC pipeline, the last convolutional layer of ResNeXt is followed by Sum pooling layer, L2-normalization layer and a PCA+Whitening layer.
- **Regional Maximum Activations of Convolutions (RMAC) [34.7% mAP]:** In RMAC, the last convolutional features of ResNeXt are max-pooled across several multi-scale overlapping regions. The region based descriptors are L2-normalized, PCA+Whitened and L2-normalized again. Finally, the descriptors are sum-aggregated into a single descriptor.

The underlying CNN networks (ResNet and ResNeXt) are pretrained on ImageNet, and fine-tuned on a subset of the Landmarks dataset used in the work of Babenko et al., that contains approximately 120k images of 650 famous landmarks sites.

The images in this dataset were originally collected through textual queries in an image search engine without thorough verification, and therefore they contain a significant number of unrelated images which we filter out and remove. This procedure is semi-automatic, and relies on dense SIFT features detected with a Hessian-affine detector and aggregated with the RVD-W descriptor. This cleaning process leaves about 25,000 images still belonging to one of the 650 landmarks, which is what we then used to fine-tune our models for this competition.

We didn't use the data from the landmark classification sister competition for training, as we wanted to see how well our solution would generalise to new dataset (as opposed to creating a solution which was specifically tailored to this dataset).

Combined Descriptor

Our final global descriptor was built by concatenating six fine-tuned global descriptors trained as explained above (LB score shown in brackets):

- ResNeXt+REMAP (**42.8%**)

- ResNeXt+RMAC (34.7%)
- ResNeXt+MAC (32.9%)
- ResNeXt+SPoC (31.7%)
- ResNet+REMAP (35.8%)
- ResNet+MAC (30.4%)

We assign each descriptor a weight by scaling them to a fixed L2 norm, and concatenate as follows:

```
XG = [2× ResNeXt+REMAP;  1.5× ResNeXt+RMAC; 1.5× ResNeXt+MAC; 1.5× ResNeXt+SPoC; ResNet+MAC; ResNet+REMAP]
```

The weights are selected ad-hoc to reflect the relative performance of each method. After this, we perform PCA to reduce the dimensionality of the descriptor to 4K (not just to save computation, but also to discard noisy dimensions) and apply whitening so that all dimensions have equal variance. While PCA and whitening only gave a small improvement on its own, it improved the result of our query expansion step by several %.

Nearest Neighbour Search

After descriptor creation, each image is represented by a 4096-dimensional descriptor. Next, we use an exhaustive k-nearest neighbour search to find the top 2500 neighbours and L2 distances for each image – nothing fancy here. Submitting the top 100 neighbours for each test image at this stage nets a score of **47.2%**.

This step is implemented using optimised NumPy code and takes 2 hours to find the top 2.5K neighbours for each of the 1.2M images (we needed the nearest neighbours for the index set too for the next steps).

Database augmentation

The next step is to perform database-side augmention (DBA), which replaces every image descriptor in the database (including queries) with a weighted combination of itself and its top 10 neighbours. The objective is to improve the quality of the image representations by leveraging the features of their neighbors. More precisely, we perform weighted sum-aggregation of the descriptors, with weights computed using:

```
weights = logspace(0, -1.5, 10)
```

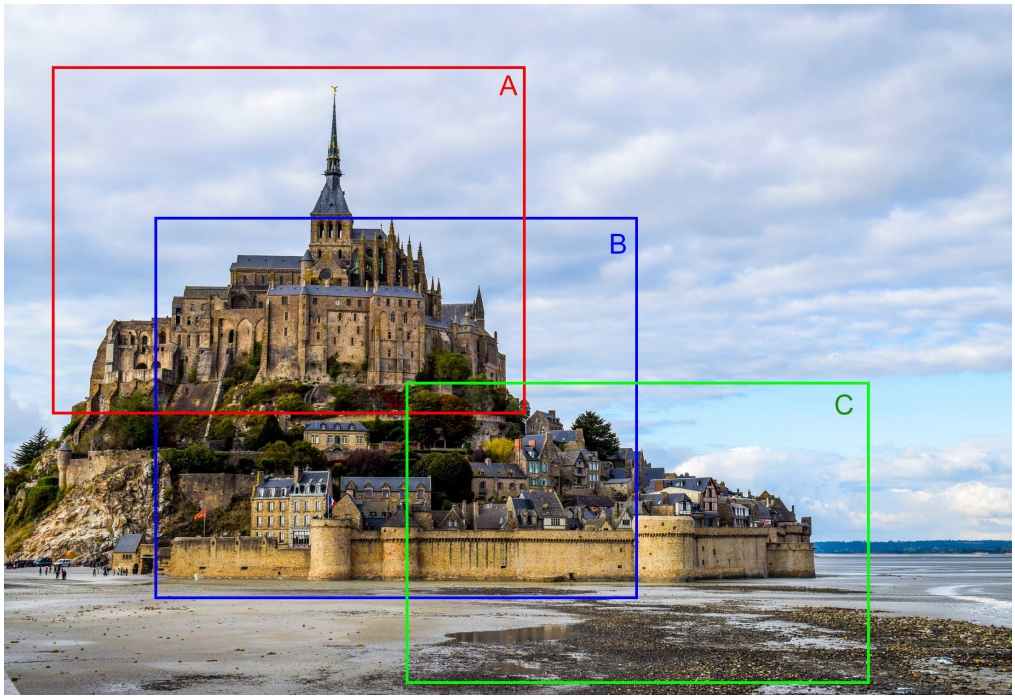
Interestingly, while on other datasets we find that using more than a couple neighbours for augmenting the *query* is detrimental to the score, we found 10 neighbours to be the optimum for both query and database images for this dataset.

It should be noted that DBA was the last thing we added to the pipeline, and while it gave a huge improvement to our score on its own, when combined with the query expansion step, the improvement is only a modest 1-2%. We believe this is because the DB expansion acts very similar to the first level of our query expansion method.

Query Expansion

Query expansion is a fundamental technique in image retrieval problems which often nets a significant improvement in performance. It works off the principle that if images A <->

B, and B <-> C, then A <-> C (even if descriptors A and C are not explicitly matched). A simple example where the benefit of this is illustrated is in cases where there are three partially overlapping images:



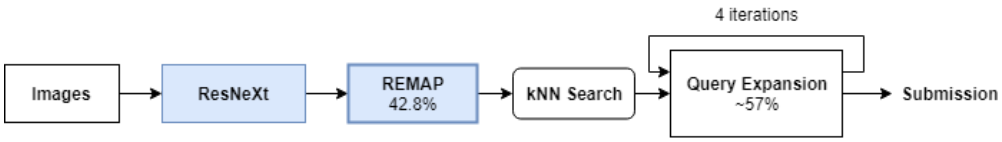
In this case, a query expansion scheme can help us match images A and C as being of the same scene, even though the descriptors themselves (especially global descriptors) are very unlikely to match. This also benefits us in other cases where, for example, we have several images with gradually different perspectives or lighting conditions, and intermediate images can help us to connect them.

In this competition, we designed a novel, fast technique for query expansion which can be run recursively to capture long-distance connections between images with many intermediate steps, which lends itself well to this problem as some landmarks have very many images - only one image has to be matched to a query and all the images of the same landmark are then also appended to the top 100 list.

We've opted not to discuss the exact method here and instead explain it in-depth in a paper in the coming weeks. But the basic principle is the same as other well-established query expansion techniques. Our query expansion gave an approx **11%** improvement when run for one iteration, and **14%** improvement when run recursively (with 30min runtime) - this is reduced when combined with DBA.

Simplified Model

In a production ML environment, it's rare to use a monster ensemble of various models when the laws of diminishing returns apply - Usually most of the performance can be achieved with a small subset of the complexity. We expect a slimmed-down version of our solution with just ResNeXt-REMAP and query expansion to score 56-57% with <12 hours total runtime from downloading the data to submission (on 4 GPUs) - largely aided by the fact that it did not have to be trained for this competition.



Things that didn't work

- **Local descriptors:** This was perhaps the most surprising thing for us in the competition. We tried several schemes based on various local descriptors, with or without geometric verification, for example using it for reranking our results (decrease in performance), or using it to scan through the top few thousand global neighbours to find very confident local matches that the CNN-based globals missed (helped, but the improvement was sub-0.1%). We're very curious if/how other teams were able to harness the power of local descriptors in this competition. Perhaps CNN-based global descriptors are so good now that the era of locals is over?
- **Coping with rotated images:** A glance through the dataset showed us that a significant number of images were rotated. We tried accounting for this in multiple ways, for example by also comparing rotated and non-rotated descriptors during kNN and taking the closest match for each pair of images, however we could not increase our score. This is probably because as this is a dataset with very many distractors, the increase in false positives outweighs the gain in true positives.
- **Ensembles:** We tried various methods to combine very different submissions with methods such as rank averaging and interleaving predictions, however the gains we saw from this were very small. It seemed to work better to combine models earlier on within the solution than at the end.

Acknowledgments

Components of our solution have been supported by work done under the industry project "iTravel - A Virtual Journey Assistant", InnovateUK grant #102811 (funder: UK Research and Innovation).
As this is an industry-led project, we are unfortunately unable to release the code for our solution under the Apache 2.0 License (which includes royalty-free commercial use).

Finally, I would like to say thank you to Kaggle and the organizers for organising this very interesting competition - it is unusual and refreshing to see both retrieval and semi-supervised learning competitions on Kaggle. I would also like to thank my excellent teammates Sameed and Mirek for their hard work and fun moments and everyone at both CVSSP and Visual Atoms for tracking our progress and supporting throughout. The whole competition has been a very enjoyable experience for us.

We will do our best to answer any questions you might have about our solution, and we look forward to seeing the solutions of the other teams :)

- Mikel (with the team)
CVSSP & Visual Atoms

Options

Comments (54) Sort by Hotness

Click here to enter a comment...

Ilya Ploskovitov • 15 days ago • Options • Reply

1

congrats on winning the competition!

aixili2014 • (11th in this Competition) • 18 days ago • Options • Reply

3

Congrats! Thanks for your kindly sharing!

The underling idea is similar to a paper published in ECCV2016 and IJCV, i.e. 'Deep Image Retrieval: Learning global representations for image search'(DIR)

We extract the RMAC feature by the public model released in this paper, which consumes about 40 mins to process about 10k images in a GTX 1060 GPU. Each image is represented as a 2048-d feature, and the database feature is about 8GB. For the basic search, we use a desktop computer with 32GB memory to compute cosine similarity.

The database side augmentation is performed by the knngraph, which is fast but may not accuracy as finding the neighbors by computing the distance between each database images. Our implementation only improves about 2 mAP after database side augmentation.

anokas Topic Author • (1st in this Competition) • 17 days ago • Options • Reply

0

Simple but effective!

CoMartel • 18 days ago • Options • Reply

3

Congrats for this not so easy win! I'm curious about the hardware your needed for this competition : what did you use ?

anokas Topic Author • (1st in this Competition) • 18 days ago • Options • Reply

5

Good question.

Most of the work was done on two machines: one with 4 Pascal GPUs and another dual xeon machine (52 cores/768GB RAM, although we didn't need more than 200). At times we had access to more machines temporarily (mostly CPU).

Eric • (92nd in this Competition) • 18 days ago • Options • Reply

0

Ok, I am a small player with my 8 Cores then and 16 Gb RAM.

CoMartel • 18 days ago • Options • Reply

1

ok, so nothing too fancy ^^. Thanks ;)

Sanyam Bhutani • 18 days ago • Options • Reply 3

Congratulations @anokas! You're an inspiration for all of us :D

(PS, I had bet my friends you'd win-Thank you for that :P)

Daniel Dopp • (109th in this Competition) • 16 days ago • Options • Reply 1

Hi guys, congratulations on the win! It was a lot of fun watching the top teams fighting it out on the leaderboard. This is my first Kaggle competition and I'm wondering how you all handled processing time so well.

I took a similar (albeit significantly less sophisticated) approach of using a pretrained network to generate global features and followed that up with similarity comparisons to find the 100 closest neighbors to each test image. However when I tried doing an initial (naive n^2) similarity comparison I had a computation time estimate ranging from 4 months on a parallelized 8 core CPU to about 1 month on 2 1080ti GPU's. Even when using a smarter KD tree based nearest neighbor search my time to run a full submission is about 5 days. Not to mention the time taken for one off things like initial feature generation.

It seems that you all got your pipeline running in less than a days time from start to finish though. Is this attributed just to beefy hardware or did you do anything clever to cut down on computation time?

anokas Topic Author • (1st in this Competition) • 16 days ago • Options • Reply 1

> However when I tried doing an initial (naive n^2) similarity comparison I had a computation time estimate ranging from 4 months on a parallelized 8 core CPU to about 1 month on 2 1080ti GPU's. Even when using a smarter KD tree based nearest neighbor search my time to run a full submission is about 5 days. Not to mention the time taken for one off things like initial feature generation.

This is what I also had initially. I spent some time optimising this n^2 similarity comparison and it now takes about 1 hour for the whole test set on 8 cores, or 12 hours for the whole index+test set (1.2M^2 comparisons). This is entirely doable without kd-tree or similar, although our approach only works with L2 distancr. I'll post the code we used for this when I can.

Felipe Sens Bone... • (86th in this Competition) • 17 days ago • Options • Reply 1

Amazing explanation and work, congratulations! I've also used SPoC descriptors but couldn't get close to your results. How did you handle different image sizes?

SAMEED • (1st in this Competition) • 17 days ago • Options • Reply 4

Thank you. For training as well as testing, each images is resized to 1024*768 pixels. The SPoC descriptor worked well because it is trained in an end-to-end fashion on Landmarks-retrieval

dataset. We design a three stream Siamese network architecture where the image signature produced by each of the three streams are jointly considered by triplet loss. All our networks ResNext101+ (REMAP, RMAC, MAC and SPoC) are trained in this way.

JuneHaoChing • 18 days ago • Options • Reply

1

Congrats! A newbie here, just want to ask... is there any special way in separating the train,val and test sets for the model? Or the whole flow goes like image retrieval function, extract descriptor from image and recognize it..?

anokas Topic Author • (1st in this Competition) • 18 days ago • Options • Reply

2

The model can be thought of in some ways as unsupervised - we have no training data and no validation data. We used the leaderboard and also looking at various statistics of the behaviour of kNN, QE etc. to try and judge what changes to the model were good or not.

We could have created a dataset out of the data from the classification competition and use that for training/validation, but we decided not to and to instead try and create a generic solution.

I think that our stability on the private LB may be largely due to the fact that we tried to create a robust model by not tuning it specifically for this data, so changes in the data distribution had less of an effect on us than it perhaps did on others.

Andres Torrubia • 18 days ago • Options • Reply

1

Congratulation, especially for bringing novel contributions. Well done!

CPMP • 18 days ago • Options • Reply

2

Congrats! Very interesting write up, lots to learn for me.

Your query expansion reminds me of how I used transitivity in Quora. You use *A r B and B r C implies A r C*, but I found a way to also use *A r B and B not r C to imply A not r B*. I wonder if the same idea could have been used here, see my discussion at the time:

> Transitivity of is duplicate. If q1 is duplicate of q2 and q2 is a duplicate of q3 then q1 and q3 are most probably duplicates. We say most probably because this is not 100% true on the train dataset. A variant is: i If q1 is duplicate of q2 and q2 is not a duplicate of q3 then q1 and q3 are most probably not duplicates. One way to capture this is to to say that if q1 and q2 have similar properties with respect to their common neighbors, then they are probably duplicates. We therefore compute, for each level 1 prediction p, and for each row (q1, q2): the series of predictions p(q1,q) and the series of predictions p(q2,q) for all q common neighbours of q1 and q2. We then compute distances between these series, and add it as a feature for level 2. We improved this in the last day of the competition by using the target value instead of train predictions in an out of fold manner.

I'd be curious to know if you find this relevant or if I am missing the point. Full write up there, but the relevant part is what i quoted: https://www.kaggle.com/c/quora-question-pairs/discussion/34342

anokas Topic Author • (1st in this Competition) • 18 days ago • Options • Reply 4

This is an interesting point you bring up, and not something I thought of. However, I'm not sure how much finding these negative relationships would benefit here.

The competition is interesting in the sense that we have to provide top 100 matches in the training set for each query. In most queries, global descriptors may only find one or two positive matches (this is a case of high false negative rate and low false positive rate).

Looking at the distribution of distances in this dataset, it seems that the queries are very far removed from the database images (images are very tough to match), while the database images are easily matched to each other. In this case, the job of the global descriptor is to find one or two matches, allowing us to "embed" the query somewhere in the database space, and then query expansion finds the 100 closest images in the database space that have the highest probability of matching the detected matches.

However, this is not a binary matching/non-matching problem, the trick is each database image has to be assigned a probability of matching the query. It can be viewed as a graph problem with 1M nodes and 2.5B edges (top 2500 neighbours are known for each node) and you have to determine the probability of two nodes actually being connected (which is difficult given you have L2 distances and not probabilities - and trying to map them just doesn't work, we tried).

It boils down to the fact that we don't have an issue with over-predicting, but with under-predicting (in my view). So while I think your approach could definitely help add performance, it might not be huge, and would need to carefully tuned not to increase the false negative rate too much. An interesting experiment either way.

Eric • (92nd in this Competition) • 18 days ago • Options • Reply 1

On the transivity remark, if you are using cosine similarity, you will get somehow for free. Depending on the type of metrics you are using for nearest neighbours, you should get it by design. Also using some kernel nearest neighbours could somehow help.. The tricky part being to find the right kernel to transform the data in the right RKHS (Reproducing Kernel Hilbert Space). This reminds me the kernel trick for SVM.

CPMP • 18 days ago • Options • Reply 2

@Anokas, makes perfect sense that non transitivity isn't much relevant here. I guess you're computing some sort of transitive closure of the weighted graph where p(A, B) is the probability A and B match. I guess I would have started with p(A, C) = max of min(p(A,B), p(B,C)) for all B common neighbors of A and C , but I'm sure you are using something more complex as this one may lead to too many matches.

@Eric, interesting analogy with kernels for SVMs. In the Quora case, handling transitivity on top of cosine similarity was helping significantly.

old-ufo • (14th in this Competition) • 18 days ago • Options • Reply 1

This sounds similar to diffusion algorithm here https://arxiv.org/abs/1611.05113 which improves for from 0.304 base network to 0.46 with "query expansion" https://github.com/ducha-

aiki/manifold-diffusion But we haven`t trained so powerful base network, looking forward to the paper :)

hlzz • 18 days ago • Options • Reply

2

congrats on winning the challenge, well done and thanks for sharing this solution. Unfortunately I cannot attend cvpr workshop this year so I want to post some of my questions here. How many layers are there in your resnet and resnext model, 50 or 101? Have your tried combine different aggregation methods (REMAP, MAC, SPoC, etc.) with a single model, rather than on two models? If yes, which brings more boost to performance, combining different aggregation or combining more network models? Thanks.

SAMEED • (1st in this Competition) • 18 days ago • Options • Reply

1

Thank you. We used ResNet101 and ResNeXt101 CNNs. With a single model (ResNeXt101), the performance of our 4K descriptor is 61.1%. The extra 1.6% is achieved by concatenating two ResNet101 architectures. The big boost in performance is obtained by combining different aggregation method.

hlzz • 18 days ago • Options • Reply

0

Bravo! Look forward to your paper.

soywu • 18 days ago • Options • Reply

2

congrats on winning , hope to see your paper soon

Minchul Shin • (9th in this Competition) • 18 days ago • Options • Reply

2

congrats on winning the competition!! :-) I hope we can see at CVPR 2018! well done!

Eric • (92nd in this Competition) • 18 days ago • Options • Reply

0

Anokas , If I am correct, you are going to join the pinacle circle of grand masters with your fifth gold medal, isn' it.... Impressive. You are my hero! In addition, I notice in your profile that you are only 16 years old. So will you be the youngest or at least one of the youngest Kaggle Grand master?

anokas Topic Author • (1st in this Competition) • 18 days ago • Options • Reply

3

For grandmaster, you need one of those gold medals to be a solo, so not quite yet ;) I'm working on it

Thank you for the kind words though!

Eric • (92nd in this Competition) • 18 days ago • Options • Reply

0

Almost there. You just need one solo medal... which should not be out of reach for you. By the way, I saw that your team mates are professor. Are you doing a PhD by any chance?

anokas Topic Author • (1st in this Competition) • 18 days ago • Options • Reply

2

Nope, I am not at university yet.

Eric • (92nd in this Competition) • 18 days ago • Options • Reply

1

You are amazing to be working with professor at such a young age... And to know so much... Well done and even more impressive!

David • (69th in this Competition) • 18 days ago • Options • Reply

2

Truly excellent result and thanks for the high quality write-up, looking forward to seeing the full paper.

Eric • (92nd in this Competition) • 18 days ago • Options • Reply

0

Well done to all of the member of your team as well!

Eric • (92nd in this Competition) • 18 days ago • Options • Reply

0

Well done anokas! And thanks for the detailed explanation. Number one position well deserved

SnailFast • 18 days ago • Options • Reply

0

Well done !

Abhik Banerjee • 17 days ago • Options • Reply

0

Congratulations to the team on your great achievements, am just curious to learn if there are any kernels that share some of the implementation ? or some of the partial implementations ?

SamDesai • 16 days ago • Options • Reply

0

Awesome

15 days ago

This Comment was deleted.

Mohan • 15 days ago • Options • Reply 0

Great work

Zahid Ahmed • 15 days ago • Options • Reply 0

Wow ! Great Job Guys

Aman Srivastav • 14 days ago • Options • Reply 0

Congratulations on your win :) The post is awesome!

Arya Prabhudesai • 14 days ago • Options • Reply 0

Congratulations!

Christian Vogel • 13 days ago • Options • Reply 0

congrats !

13 days ago

This Comment was deleted.

hero576 • 13 days ago • Options • Reply 0

Congratulations!

Gabriel Melo de ... • 13 days ago • Options • Reply 0

Nice work, well done!

Markus Heimerl • 12 days ago • Options • Reply 0

Great solution!

Sky Zhou • 10 days ago • Options • Reply 0

Congrats and thank you for sharing!

Sergio Fraile • 10 days ago • Options • Reply

0

This is an amazing post, thanks and congrats on that well deserve victory!

Eikma Rizal • 10 days ago • Options • Reply

0

Congrats and thanks for the write up.

Fritjof Wolf • 8 days ago • Options • Reply

0

Great work, congratulations!

Marcos Rabaioli • 7 days ago • Options • Reply

0

Great work, very good!

NeelambujChatu... • 7 days ago • Options • Reply

0

Hi, Congrats on winning the competition.

Rajesh A • 5 days ago • Options • Reply

0

congrts anokas for winning the competition.

PiceChen • 3 days ago • Options • Reply

0

Congrats!