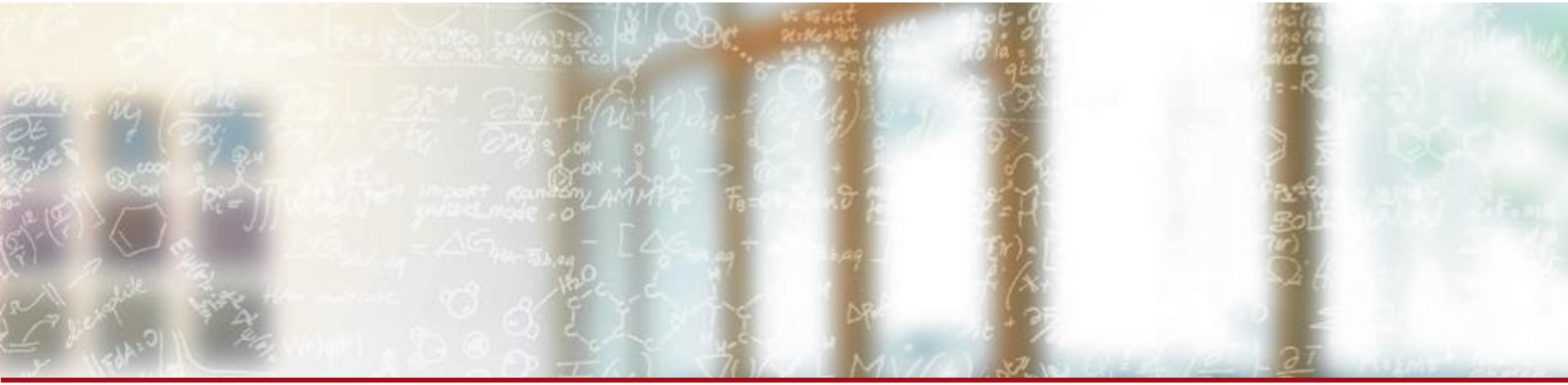


Multi-Node Distributed Training with tf.keras

Summer School 2020
Henrique Mendonça, CSCS
July 24, 2020

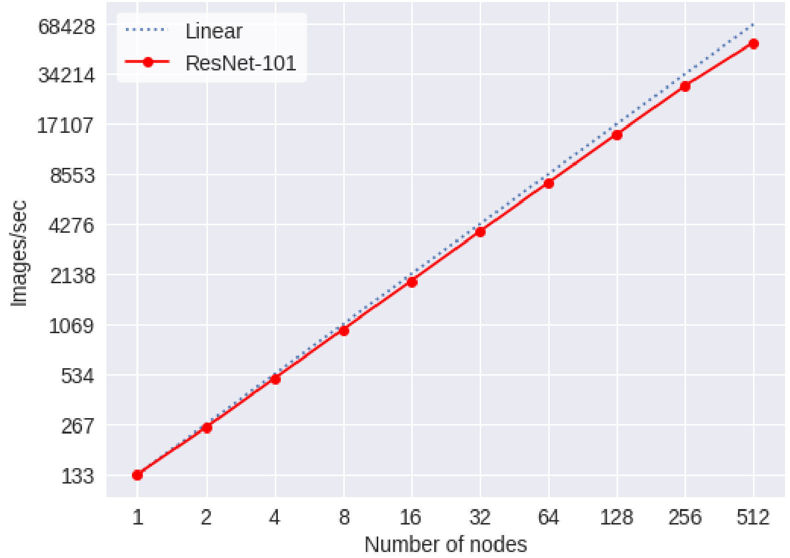
Table of Contents

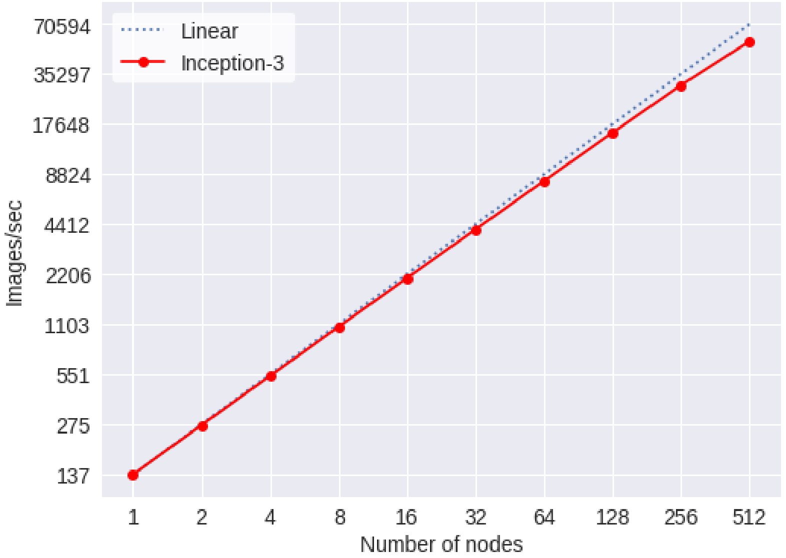
Motivation

- Why should we use multiple GPU's to train a DL model?

Motivation

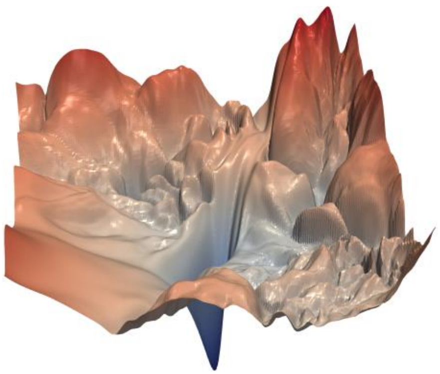
- Because it's faster
  - Allow quick experimentation of new ideas and models





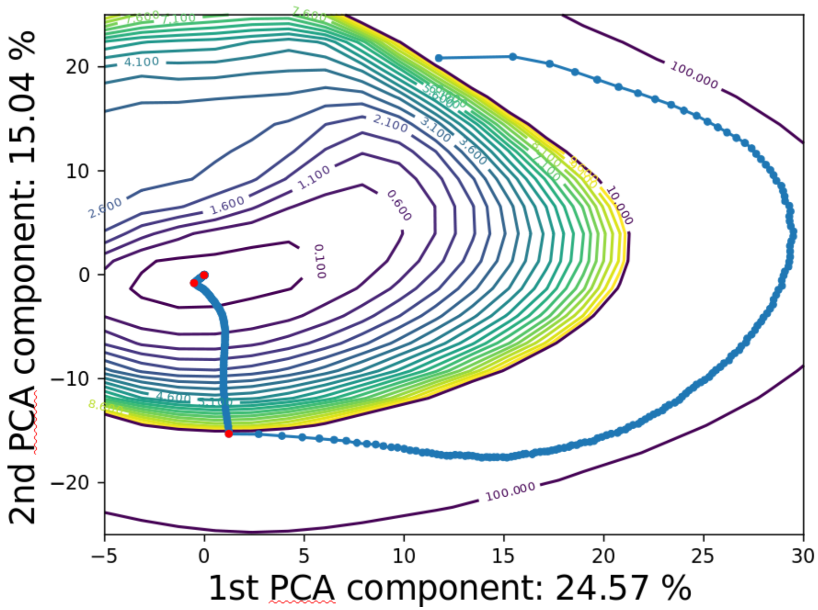
- Allows training models larger than memory
  - Out-of-Core learning: Not covered in this course
- May allow better accuracy, especially in larger models. Why?

Motivation

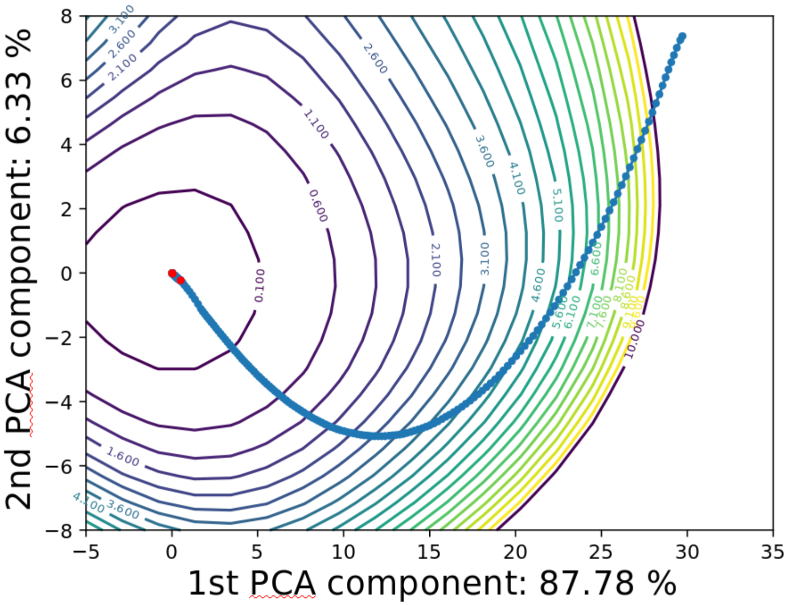


Visualizing the Loss Landscape of Neural Nets
https://arxiv.org/pdf/1712.09913.pdf

Batch size 128

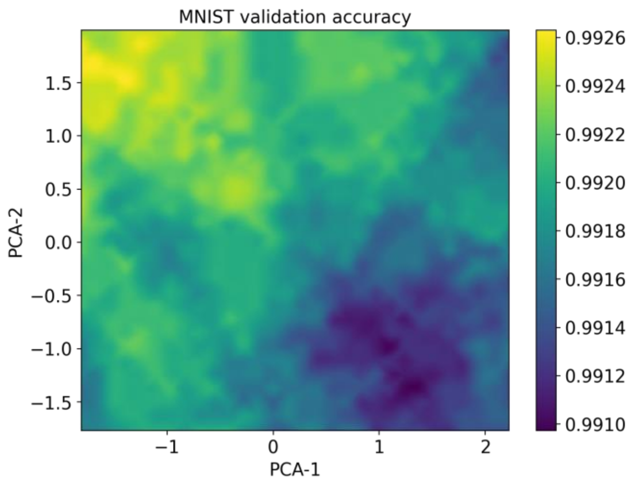


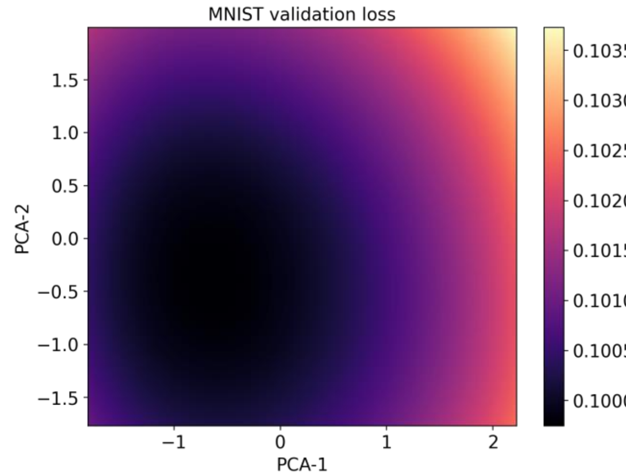
Batch size 8192



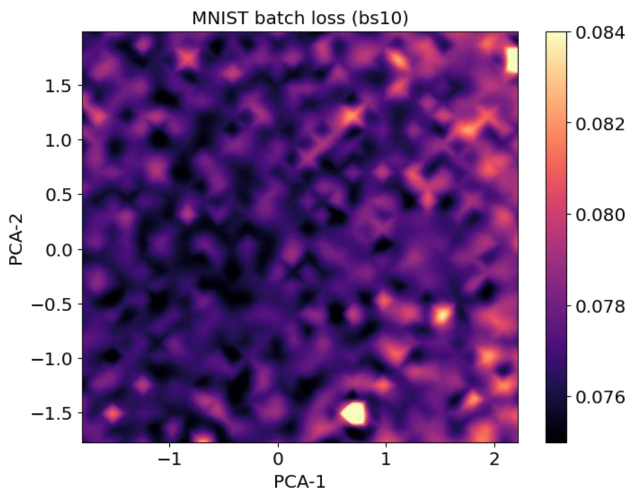
Motivation

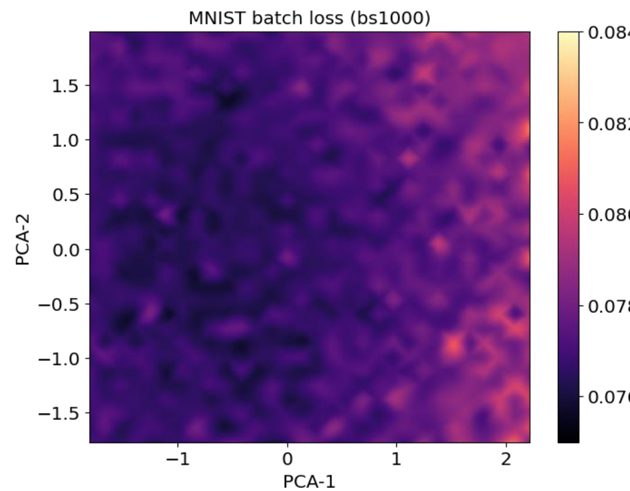
- Validation:





- Training:





Distributed training with TF Keras 2.2

- TensorFlow introduces tf.distribute.Strategy
  - Wraps model in a distributed scope

- Multi-GPU -> tf.distribute.MirroredStrategy
  - Single node/worker

```
strategy = tf.distribute.MirroredStrategy()
with strategy.scope():
   model = build_and_compile_model()
model.fit(dataset, epochs, steps_per_epoch)
```

  - NCCL AllReduce by default
  - Automatic data sharding across GPU's

Multi-worker Distribution

- Creates some additional complexity
  - external network communication
  - separated OS
  - separated processes
    - Facilitated by ipyparallel magic on JupyterLab

Multi-worker Distribution

- tf.distribute.experimental.MultiWorkerMirroredStrategy
- Communication:
  - NCCL AllReduce for all-reduce (if available)
  - Ring algorithm for all-gather
  - Includes fault tolerance when using ModelCheckpoint*
- Cluster Resolver:
  - defaults to TFConfig

```
os.environ['TF_CONFIG'] = '{
    "cluster": {"worker": ["nid01111:8888", "nid02222:8888"]},
    "task": {"type": "worker", "index": "0"}
}'
```

** TF 2.3 uses the BackupAndRestore callback instead*

Multi-worker distribution with SLURM

- TensorFlow 2.2+

```
tf.distribute.cluster_resolver.SlurmClusterResolver(
    port_base=8888, auto_set_gpu=True, rpc_layer='grpc',
    jobs=None, gpus_per_node=None, gpus_per_task=None,
    tasks_per_node=None
)
```

- All parameters are automatically queried from SLURM

Multi-worker distribution with SLURM

```
%%px
strategy = tfd.experimental.MultiWorkerMirroredStrategy(
    cluster_resolver=tfd.cluster_resolver.SlurmClusterResolver(),
    communication=tfd.experimental.CollectiveCommunication.NCCL,
)
with strategy.scope():
    model = build_and_compile_model()
model.fit(dataset, epochs, steps_per_epoch)


# Done!
```

Multi-worker distribution with SLURM

- Practise
  - Run MNIST training and inference on 2 GPU's

Batch Norm Synchronisation

- How does Batch Normalisation work?
- What happens if it's used with small batch sizes?

Batch Norm Synchronisation

- BN: Exponential Moving Average per channel

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \epsilon}}$$

$$y_i^{(k)} = \gamma^{(k)}\hat{x}_i^{(k)} + \beta^{(k)}$$

- tf.keras.layers.experimental.SyncBatchNormalization
  - AllReduce across BN layers during forward pass
  - Synchronizes all statistics before autodiff

Batch Norm Synchronisation

- Practise
  - try [tf.keras.layers.experimental.SyncBatchNormalization]
  - Should we see any accuracy improvement in MNIST?

Scaling learning rate and momentum

- Linear scaling LR
  - One weird trick for parallelizing convolutional neural networks
    https://arxiv.org/pdf/1404.5997.pdf
  - Accurate, Large Minibatch SGD:Training ImageNet in 1 Hour
    https://arxiv.org/pdf/1706.02677.pdf

- Square root scaling LR
  - Large Batch Optimization for Deep Learning: Training BERT in 76 minutes
    https://arxiv.org/pdf/1904.00962.pdf (LAMB optimizer extends LARS)

- Momentum:
  - Smooths out the error surface
  - Very large batches might not require momentum
    - e.g. RMSprop instead of Adam

Assignment

Kaggle: SIIM-ISIC Melanoma Classification

Identify melanoma in lesion images

(Final submission deadline: **August 17, 2020**)

kaggle.com/c/siim-isic-melanoma-classification

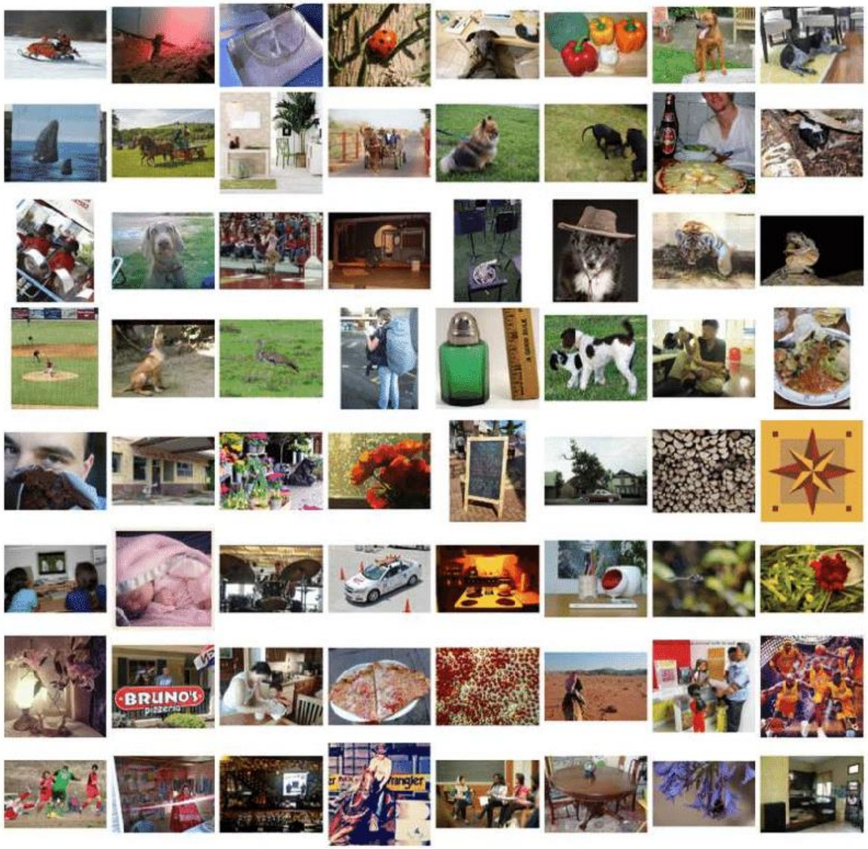
Dataset - SIIM-ISIC Melanoma Classification

	image_name	patient_id	sex	age_approx	anatom	diagnosis	target
0	ISIC_2637011	IP_7279968	male	45	head/neck	unknown	0
1	ISIC_0015719	IP_3075186	female	45	upper extremity	unknown	0
2	ISIC_0052212	IP_2842074	female	50	lower extremity	nevus	0
3	ISIC_0068279	IP_6890425	female	45	head/neck	melanoma	1
4	ISIC_0074268	IP_8723313	female	55	upper extremity	unknown	0
...	...	...	...	...	...	...	...
33121	ISIC_9999134	IP_6526534	male	50	torso	unknown	0
33122	ISIC_9999320	IP_3650745	male	65	torso	melanoma	1
33123	ISIC_9999515	IP_2026598	male	20	lower extremity	unknown	0
33124	ISIC_9999666	IP_7702038	male	50	lower extremity	unknown	0
33125	ISIC_9999806	IP_0046310	male	45	torso	nevus	0
33126 rows							

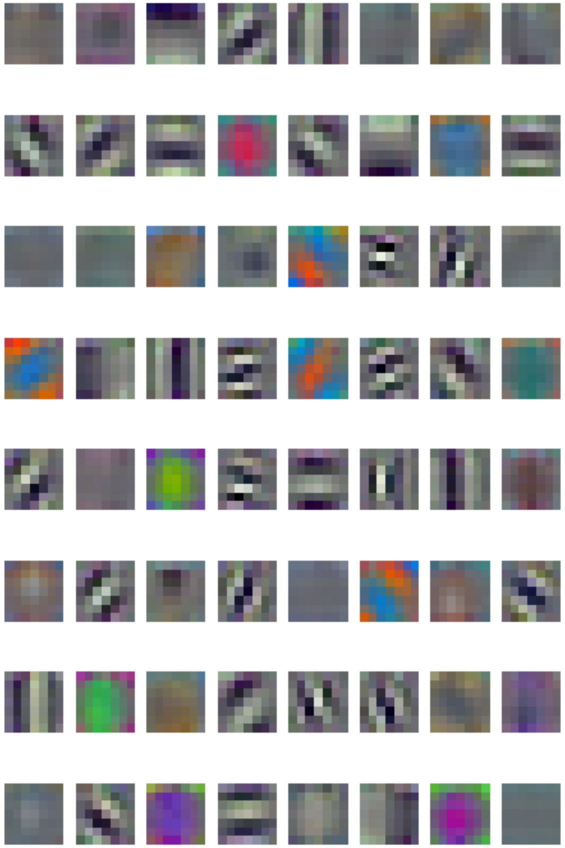
SIIM-ISIC Melanoma Classification

- Transfer Learning made Deep Learning accessible

ImageNet-2012: 1,281,167 samples



1st convolutional layer of SeResNeXT101.32x4d (ImageNet)

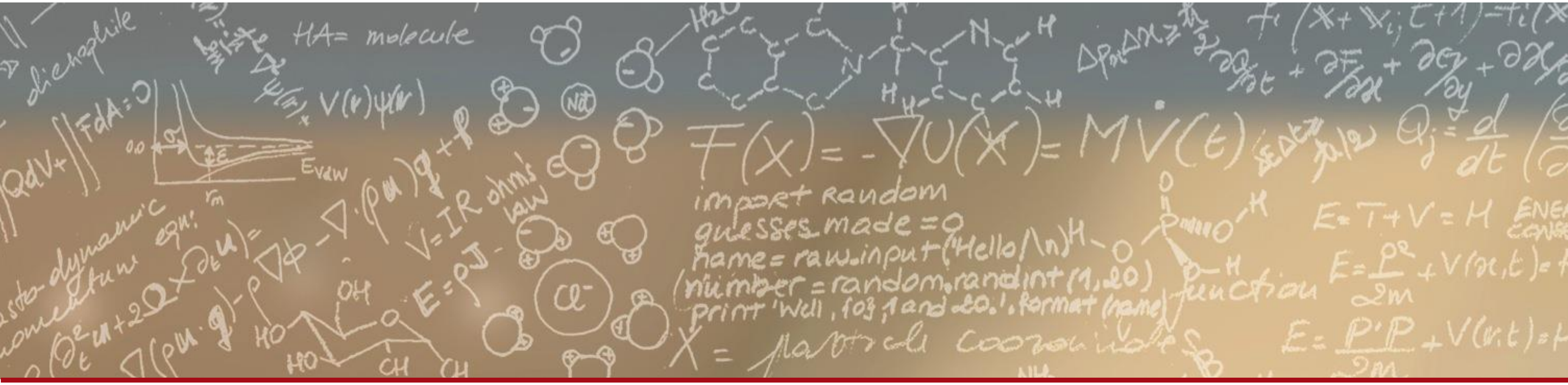


SIIM-ISIC Melanoma Classification

- Practise
  - Train on your 2 GPU's
  - Experiment with hyper parameters
  - Submit to competition

SIIM-ISIC Melanoma Classification

- Homework
  - Try Kaggle and Colab's free TPU's
    - tf.distribute.experimental.TPUStrategy
  - Read competition discussion
    - Understand the problem and metric
    - Try-out your own ideas
    - Ensemble model predictions
    - Earn a Kaggle competition medal 🏅



Thank you for your attention.