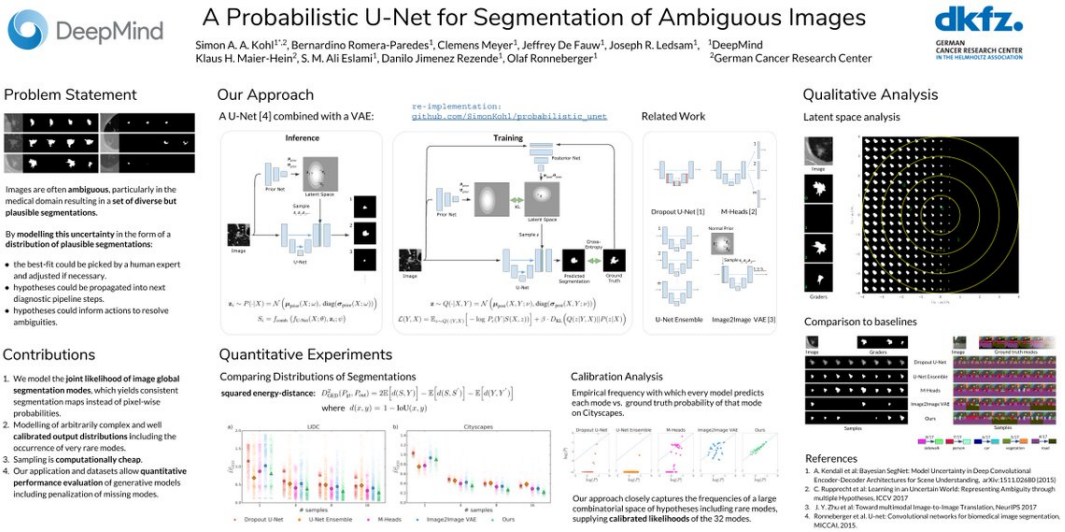
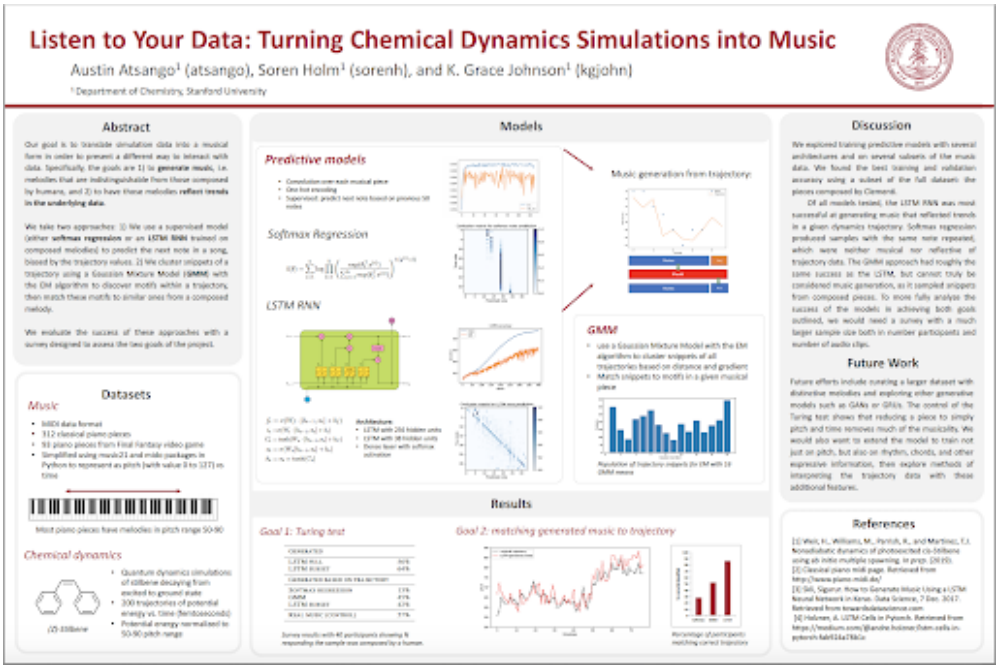
Proyecto Final

Objetivos

- Implementar los 3 modelos de deep learning más usados(de forma separada e independiente):
    - Red neuronal feed forward/MLP: datos estructurados
    - Red Neuronal Convolucional: datos espaciales
    - Red Neuronal Recurrente: datos secuenciales/temporales.
- Aplicar conocimientos del curso de investigación :
    - para la selección de datasets adecuados para DL e ideas iniciales de arquitecturas a usar(de preferencia no usar datasets usados en el curso anteriormente ya que son datasets que funcionan bien con modelos lineales y no necesitan DL)
    - Para referencia o punto de partida de modelos y arquitecturas similares según el problema y dataset elegido.
    - Se hará una presentación en formato de poster: https://guides.nyu.edu/posters





- El proyecto debe estar acompañado de un documento en formato artículo(paper) científico usando la siguiente plantilla: https://www.overleaf.com/latex/templates/neurips-2021/bfjnthbqvhgs
  - **Proyectos destacados que lo ameriten podrán ser enviados a conferencias científicas internacionales(**con ayuda de los catedráticos)

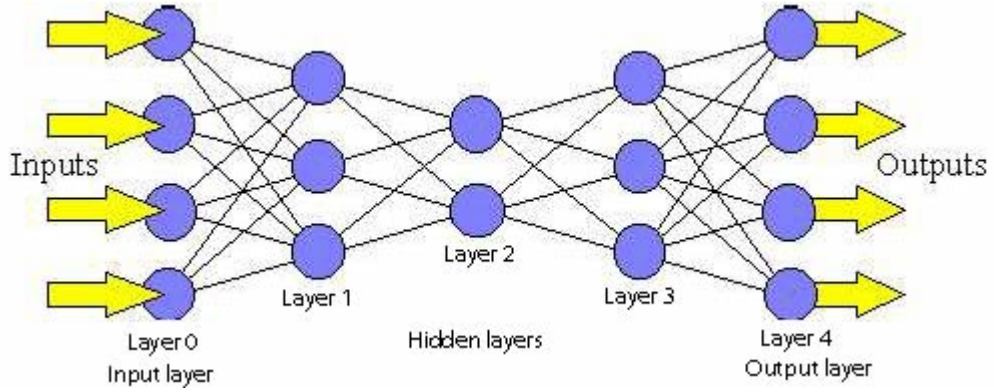
- Aplicar conocimientos del curso de algoritmos en DS para el uso de optimizadores más efectivos que gradient descent(momentum,Nesterov,adagrad,adam,etc):
  - https://github.com/janishar/mit-deep-learning-book-pdf/blob/master/chapter-wise-pdf/%5B13%5Dpart-2-chapter-8.pdf
  - http://ruder.io/optimizing-gradient-descent/

El proyecto estará dividido en 3 partes independientes entre sí(aunque es posible combinarlos y cubrir 2 partes en un único modelo para algunos casos), cada una correspondiente a un tipo de modelo de Deep Learning (puede trabajarse en 3 notebooks o scripts diferentes,cada uno puede ser usando herramienta o framework diferente)

1. **Feed Forward Network:** La primera parte consiste en una red neuronal tipo feedforward(multi-layer-perceptron)
2. **Convolutional Network:** La segunda parte consiste en una red neuronal convolucional para datos en distribución espacial(por ejemplo imágenes, o audio)
3. **Recurrent Neural Network:** La tercera parte consiste en una red neuronal tipo recurrente para datos temporales y secuenciales(por ejemplo texto,series de tiempo)
4. **(Opcional)** : Modelos de deep learning adicionales a los 3 anteriores tendrán puntos extra, por ejemplo:

a. Autoencoders
b. Attention based(transformers, etc)
c. GANs
d. semi-supervised learning
e. Graph Models

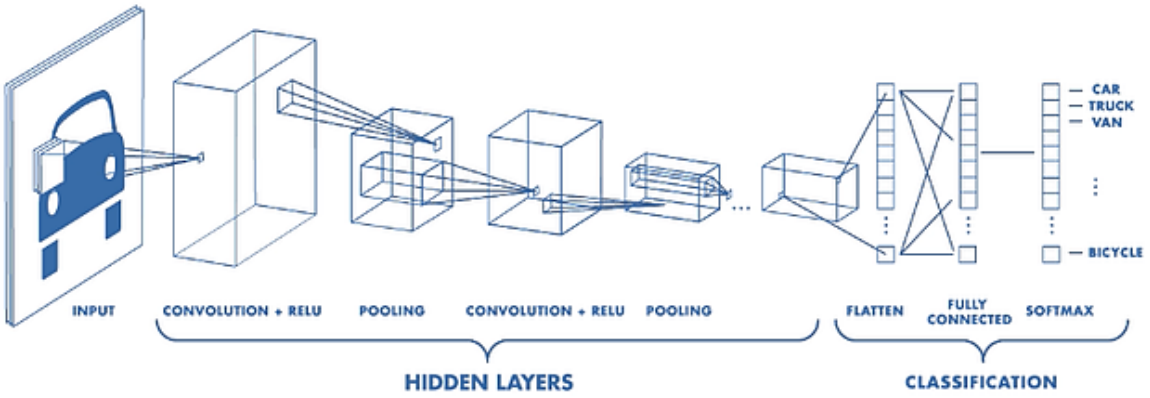
Parte 1:



La parte 1 es semi-libre, cada quien elige un dataset o problema a trabajar y entrena una o varias redes neuronales(convencionales o feed-forward) para aproximar una función que modele la relación entre ciertas variables independientes y una o más variables dependientes.

Se recomiendan datasets estructurados(organizados tabularmente en filas por observación y columnas por variable). Se recomienda: regresión o clasificación

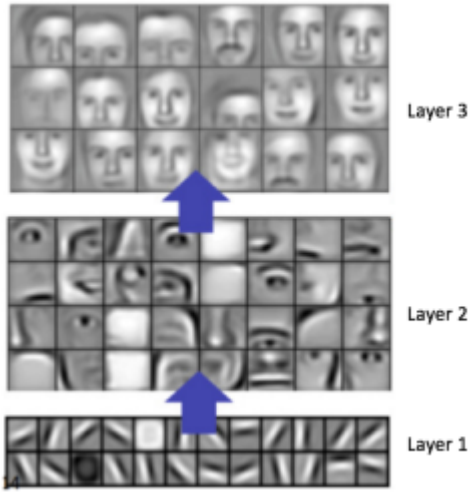
Parte 2:



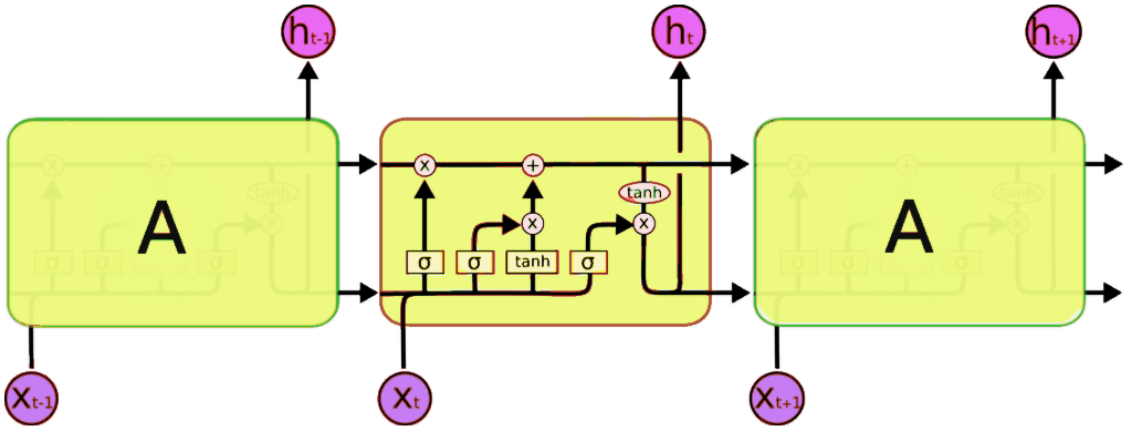
Entrenar una o varias redes neuronales convolucionales para trabajar con datos en distribución espacial(por ejemplo los casos vistos en clase: imágenes, audio, etc) .
En caso de trabajarse con imágenes:

- Deben usarse imágenes a color(no blanco y negro)
- En caso de hacer clasificación, deben ser al menos 5 posibles clases.
- No usar MNIST, CIFAR, IMAGENET o datsets de ejercicios y ejemplos introductorios (o derivados)
- Se recomienda alguno de los siguientes tipos de problemas:
    - clasificación:https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d
    - Deteccion de objetos: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088
    - Segmentacion semantica: http://www.robesafe.uah.es/personal/eduardo.romera/pdfs/Romera17iv.pdf
    - Audio: https://medium.com/@CVxTz/audio-classification-a-convolutional-neural-network-approach-b0a4fce8f6c

Se debe graficar los filtros y/o activaciones para alguna(o varias) capas convolucionales del modelo, por ejemplo:
https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a4883441533b



Parte 3:



Entrenar una o varias redes neuronales recurrentes(GRU,LSTM,bidireccional, etc) para trabajar con datos en distribución temporal/secuencial , por ejemplo :

- Sentiment analysis. https://towardsdatascience.com/application-of-rnn-for-customer-review-sentiment-analysis-178fa82e9aaf
- Detección de Spam
- Entity recognition https://link.springer.com/article/10.1186/s13321-018-0313-8
- Reconocimiento de voz. https://www.researchgate.net/publication/258818168_Speech_Recognition_with_Deep_Recurrent_Neural_Networks
- Predicción y/o generación de música. https://arxiv.org/pdf/1612.01010.pdf
- Traducción por máquina
- Chatbots

En caso de hacer NLP usar word2vec embeddings (con semántica), por ejemplo :

- **https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa**

- https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
- https://nlp.stanford.edu/projects/glove/

En caso de no usarse texto investigar o profundizar con las lecturas de la clase, cómo se entrenan los modelos de embeddings(word2vec) :CBOW ,skip-grams y explicarlo en el proyecto.

Checkpoints y deployment

Ya que deep learning requiere mayor tiempo de cómputo durante el entrenamiento se debe usar checkpoints que permitan restaurar el entrenamiento para no iniciar desde 0 en caso de fallas, cada tecnología o herramienta posee su propia forma de generar checkpoints por lo cual se debe investigar como hacerlo según la herramienta elegida(por ejemplo keras lo realiza por medio de callbacks).

Una vez elegidos los modelos finales , el checkpoint correspondiente puede ser usado para deployment(similar al proyecto del curso de ML 1),para este proyecto no se debe hacer deployment pero si se debe especificar el modelo elegido ,explicar el proceso de deployment en la herramienta elegida y subir el checkpoint asociado al repositorio de git.

Requisitos y restricciones:

- Herramientas y framework libres pertenecientes al lenguaje Python(por ejemplo NumPy, Tensorflow o Pytorch) excepto sklearn (el proyecto debe reflejar la construcción de la red, selección de arquitectura, funciones de activación, y forward-propagation,NO SOLO el model.fit(x,y) de sklearn,pero si es posible usar el de Keras). **No es necesario implementar backpropagation a mano(se puede usar usar el autograd de los frameworks)**
- En caso de desear usar GPU ,es posible usar Google Colab o instalar cuda si se desea usar uno propio.(o usar otras opciones en la Nube posiblemente pagadas como AWS , google cloud o floydhub)
- Aplicar evaluación(diferentes métricas cómo f1-score y/o precisión y recall según sea apropiado), diagnósticos,train-test split, validación cruzada y selección de hyper-parámetros para lograr una exactitud de **al menos 85%** en un set de pruebas(no en el de entrenamiento)
    - El proyecto debe reflejar la experimentación ,evaluación y selección.
    - Consultar última unidad de ML1
- No es necesario aplicar feature. eng para creación de variables y transformaciones a conceptos y features de negocio, pero si debe aplicarse conversión de categóricas a numericas(embeddings,one-hot encoding) , normalización ,tratamiento de nulls y similares.
- Usar mini-batch gradient descent(siendo el mini-batch size un parámetro)
- Aplicar conceptos “recientes” de deep-learning como parte de la experimentación:

    - Inicialización aleatoria con heurísticas(como Xavier initialization)
    - Experimentación con funciones de activación como ReLu y leaky Relu
    - Batch-Normalization : https://github.com/llealgt/Batch_Normalization/blob/master/Batch_Normalization_Lesson.ipynb
    - Regularización como Dropout
    - Optimizadores diferentes a GD, por ejemplo: Momentum ,Nesterov,RMSProp,Adagrad,Adam : http://ruder.io/optimizing-gradient-descent/
- Dejar constancia de la experimentación,evaluación y selección final de modelos.
    - Ver sección de checkpoints y deployment
- Aplicar enfoque de investigación científica:
    - Formular hipótesis respecto al comportamiento esperado para cierto experimento, capturar resultados experimentales (por ejemplo con TensorBoard,archivos de bitácora etc) y concluir basado en los resultados.

Selección de datasets

Puede usarse cualquier dataset público o propio(recopilado personalmente) , de preferencia no usado previamente en este curso o el anterior para ejemplos o prácticas(los usados previamente funcionan bien con modelos lineales y no requieren deep learning) se recomienda consultar sitios y bases de datos académicas orientadas a investigación como Arxiv , Nature , o herramientas como google dataset search:

- https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research
- https://archive.ics.uci.edu/ml/index.php
- https://toolbox.google.com/datasetsearch
- https://lionbridge.ai/datasets/20-best-image-datasets-for-computer-vision/
- https://www.kaggle.com/datasets?tagids=13207

Enviar por correo los datasets elegidos para cada parte del proyecto(descripción y/o enlace) con copia al auxiliar previo a empezar a desarrollar el modelo,explicando cómo se planea usar el dataset (para validar que sea un proyecto adecuado para ML y que no es resoluble bajo técnicas de programación convencionales).

De preferencia usar datasets específicos de situaciones o problemas de interés propio y/o nacional, por ejemplo:

- Para parte 2(convnets): imágenes médicas de laboratorios y hospitales nacionales.
- Para parte 3(modelos secuenciales): NLP para lenguas propias de Guatemala(por ejemplo clasificación de texto K'iche' o traducción de español a K'iche')

Proyectos usando datasets de interés propio y/o nacional poco explorados en la literatura de deep learning podrán ser enviados a conferencias internacionales(con ayuda de los

catedráticos). Ver en el GES casilla adicional de datasets e ideas(constantemente actualizada).

Fechas de Entrega:

- Entrega del proyecto en GES: 19 de septiembre
- Presentación en formato póster: por definir