Deep Learning Specialization

by DeepLearning.AI

Course #3: Structuring Machine Learning Projects

Course Site

Made By: Matias Borghi

Table of Contents

Summary **2**

Week 1 ML Strategy (1) **3**

Estructurando proyectos de Machine Learning 3

¿Por qué estrategia de Machine Learning? 3

Ortogonalización 3

¿Cómo se extrapola esto en proyectos de Machine Learning? 5

Setting up your goal 6

Métrica de evaluación de un número único 6

Distribución de los conjuntos de entrenamiento/desarrollo/testeo 8

Tamaño del conjunto de desarrollo y testeo 8

¿Cuándo cambiar el conjunto de testeo/desarrollo? 9

Comparing to human-level performance 9

Performance a nivel humano 9

Entendimiento del rendimiento a nivel humano 11

Problemas donde ML sobrepasa el rendimiento humano 11

Week 2: ML Strategy (2) **13**

Error Analysis 13

Análisis de errores 13

Mismatched training and dev/test set 16

Entrenamiento y testeo sobre distribuciones diferentes 16

Datos sintetizados artificialmente 18

Learning from multiple tasks 19

Transfer learning 19

Multi-task learning 20

Resumen: ¿Cuándo tiene sentido usar multitask learning? 21

End-to-end deep learning 21

Ventajas y desventajas del end-to-end deep learning 22

Summary

Week 1 ML Strategy (1)

Learning Objectives

- **Explain why Machine Learning strategy is important**
- **Apply satisficing and optimizing metrics to set up your goal for ML projects**
- **Choose a correct train/dev/test split of your dataset**
- **Define human-level performance**
- **Use human-level performance to define key priorities in ML projects**
- **Take the correct ML Strategic decision based on observations of performances and dataset**

Week 2 ML Strategy (2)

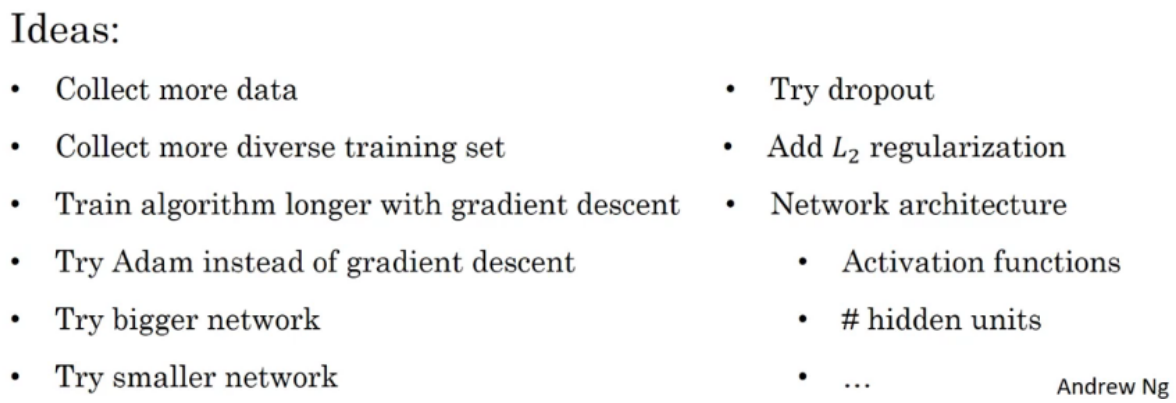
Learning Objectives

- **Describe multi-task learning and transfer learning**
- **Recognize bias, variance and data-mismatch by looking at the performances of your algorithm on train/dev/test sets**

Week 1 ML Strategy (1)

Estructurando proyectos de Machine Learning

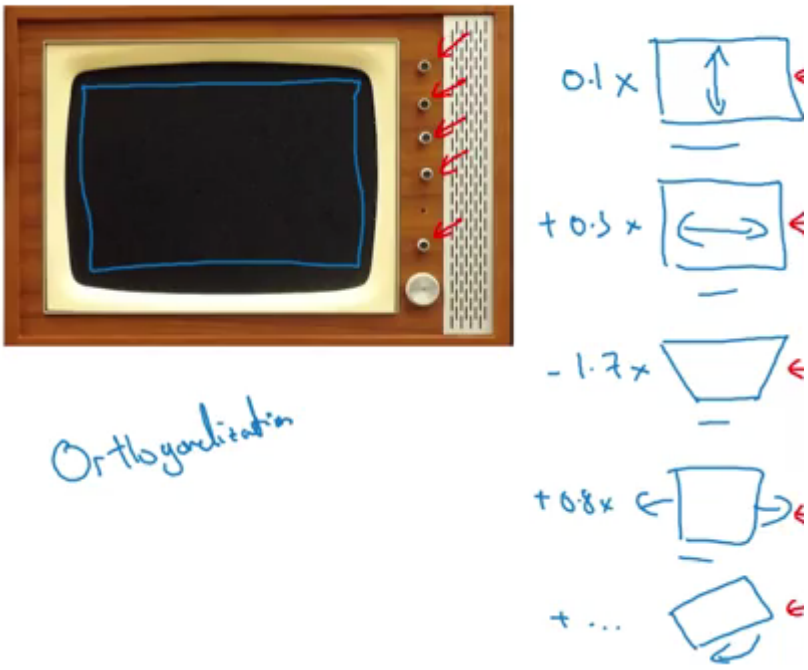
¿Por qué estrategia de Machine Learning?



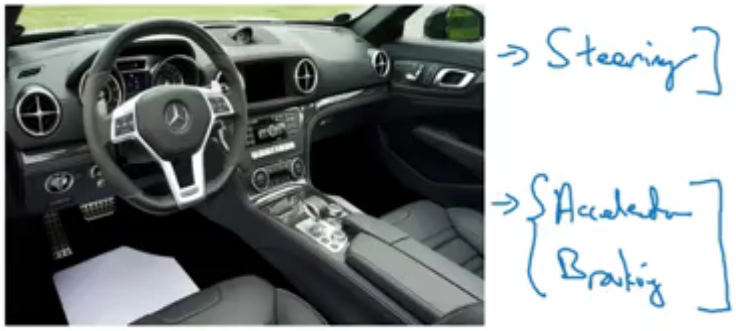
Para no perder tiempo con esto, veremos una cuestión de ideas a continuación.

Ortogonalización

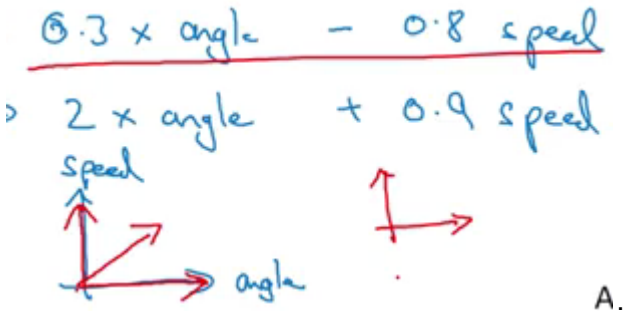
¿Qué hiper parámetro tunear para obtener un efecto deseado? Veamos un ejemplo con una televisión: Las televisiones viejas tenían unas manijas que permitían modificar por separado la altura de la posición de la pantalla, el alto, el ancho, el trapezoide, la rotación, etc. En el caso de que en lugar de esas manijas hubiera una sola que manejase 0.1 por el alto + 0.3 por el ancho, etc. la misma sería muy difícil de calibrar.



De manera similar se puede considerar un auto. El mismo posee un control para la dirección, el volante, y otros dos para la aceleración y el freno.



En lugar de estos controles se podrían definir dos como los que se muestran debajo. Pero, al ser una combinación del ángulo y de la velocidad el manejo del auto seguramente será dificultosa. Es por ello que idealmente uno querría tener un mecanismo que separe ambas contribuciones.

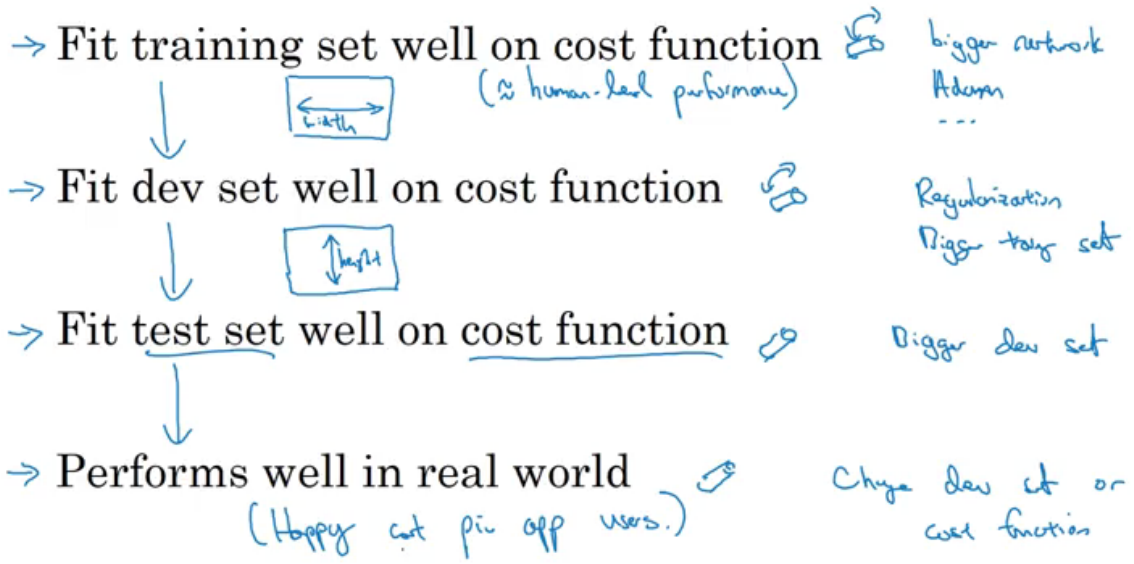


A.

¿Cómo se extrapola esto en proyectos de Machine Learning?

En ML hay cuatro pasos fundamentales:

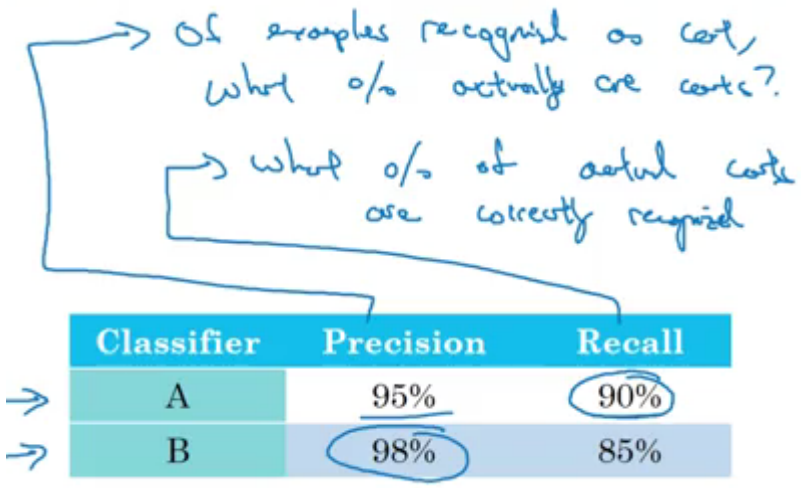
1. Ajustar el conjunto de entrenamiento con la función de coste: Si esto no se cumple se puede intentar agrandar la red neuronal o probar el Método de Adam. Esto es análogo a ajustar una perilla de la televisión, ya que hacer una de estas modificaciones mejorara más que nada el ajuste del conjunto de entrenamiento sobre la función de coste.
2. Ajustar el conjunto de desarrollo con la función de coste: Si el conjunto de desarrollo no es ajustado correctamente por la función de coste conviene implementar regularización o probar un conjunto de entrenamiento más grande.
3. Ajustar el conjunto de testeo con la función de coste: si el conjunto de entrenamiento no se ajusta correctamente a la función de coste conviene agrandar el conjunto de desarrollo.
4. Buen rendimiento en el mundo real: en este caso quizás convenga cambiar el conjunto de desarrollo o la función de coste.



Setting up your goal

Métrica de evaluación de un número único

Usualmente se suelen utilizar dos números para evaluar el rendimiento de un dado algoritmo de ML: precisión y recall.



Classifier	Precision	Recall
A	95%	90%
B	98%	85%

Precisión: de los ejemplos conocidos como gatos, que porcentaje son gatos realmente.
Recall: que porcentaje de gatos son correctamente identificados.

En este ejemplo el clasificador es mejor para el indicador Recall, mientras que el clasificador B lo hace para la precisión. El hecho de usar dos indicadores hace difícil determinar con cuál de los dos conviene seguir estudiando.

Es por ello que se recomienda utilizar una única métrica de evaluación: en la literatura de Machine Learning se suele utilizar el score F1 que combina ambas métricas.

Classifier	Precision	Recall	F1 Score
A	95%	90%	92.4%
B	98%	85%	91.0%

El mismo se define como el promedio harmónico entre la precisión y el recall.

F_1 score = "Average" of P and R.

$\left(\frac{2}{\frac{1}{P}+\frac{1}{R}}\right.$. "Harmonic mean" $)$

Supongamos ahora que se tiene la siguiente tabla donde acorde a la región se registran el error de cada clasificador. Es ideal calcular el promedio teniendo en cuenta que es un estimador aceptable y a partir de él decidir cuál de todos es el mejor. En este caso se toma al C.

Algorithm	US	China	India	Other	Average
A	3%	7%	5%	9%	6%
B	5%	6%	5%	10%	6.5%
C	2%	3%	4%	5%	3.5%
D	5%	8%	7%	2%	5.25%
E	4%	5%	2%	4%	3.75%
F	7%	11%	8%	12%	9.5%

Hay veces en las que conviene tener registro de otros parámetros para determinar con que clasificador quedarse. Consideremos por ejemplo tres clasificadores donde se registra la exactitud y el tiempo de cómputo para cada uno de ellos.

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

Se puede definir una función lineal que nos ayude a determinar con cuál de los dos quedarnos. Pero no parece la mejor manera.

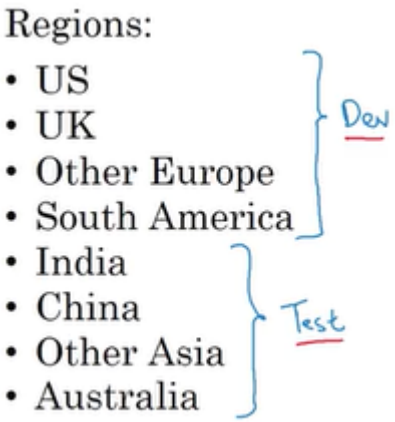
Cost = accuracy − 0.5 × Running Time

Uno podría querer maximizar la exactitud sujeto a la condición de que el tiempo de computo sea menor a 100 ms.

Distribución de los conjuntos de entrenamiento/desarrollo/testeo

Clasificación de gatos con el conjunto de desarrollo/ testeo

Suponiendo que se obtiene un conjunto de datos de varias regiones. Elegir los conjuntos de acuerdo a las regiones de las mismas es una mala idea ya que las mismas van a tener distintas distribuciones de probabilidad.



Tamaño del conjunto de desarrollo y testeo

Recién se dijo que ambos deben provenir de la misma distribución, pero, ¿cómo debe ser el tamaño de los mismos?

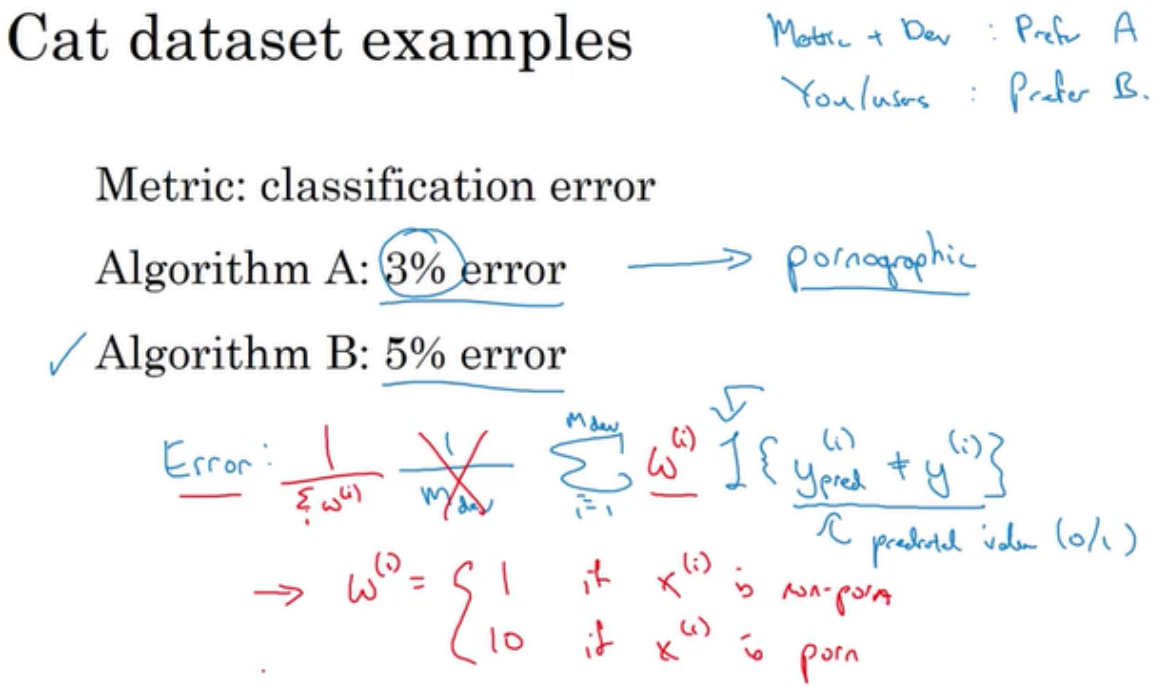
En la era del Big Data se utiliza alrededor del 98% de datos para entrenamiento.

Set your test set to be big enough to give high confidence in the overall performance of your system.

¿Cuándo cambiar el conjunto de testeo/desarrollo?

Consideremos por ejemplo dos algoritmos A y B, en el que el A posee una métrica con menor error, pero muestra contenido inadecuado siendo la preferida para el usuario.

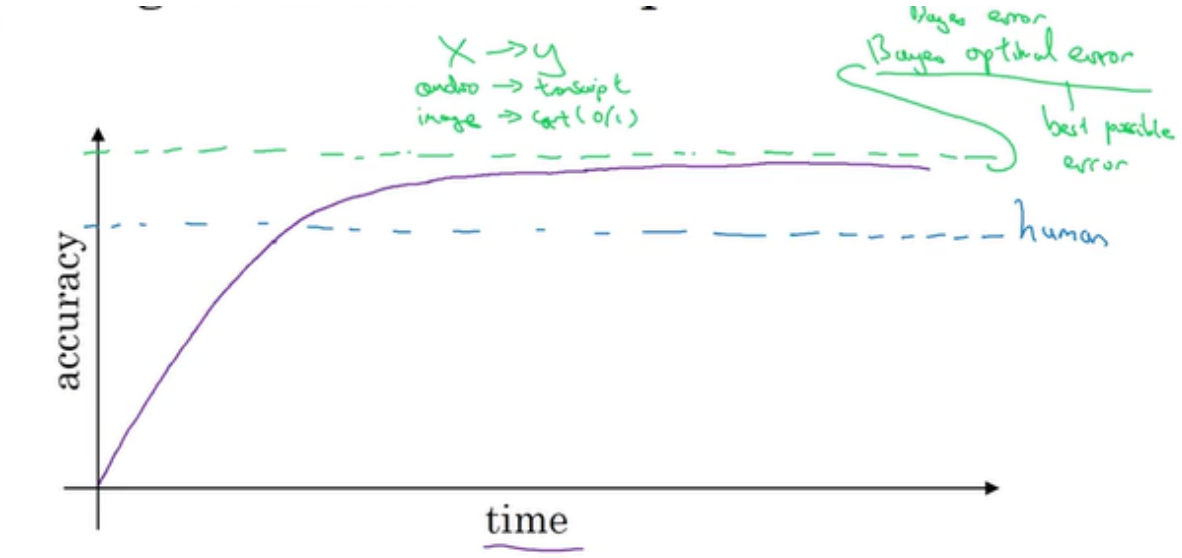
En estos casos es recomendable cambiar la métrica por la que se muestra debajo añadiendo pesos de acuerdo al contenido no deseado que aparece.



Comparing to human-level performance

Performance a nivel humano

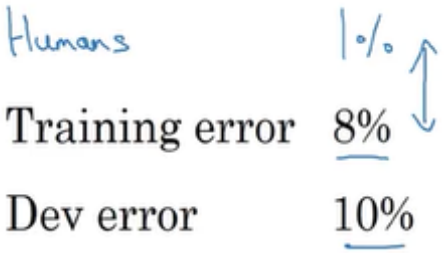
El error óptimo de Bayes es el máximo valor que podría tomar una función que mapea X en Y. Por ejemplo, dado un audio, este podría ser transcrito con una precisión máxima debido al ruido, o una imagen borrosa no sería identificable como un gato.



Existen técnicas que se pueden utilizar para llegar al nivel humano pero que no mejoran al pasar este nivel, como por ejemplo

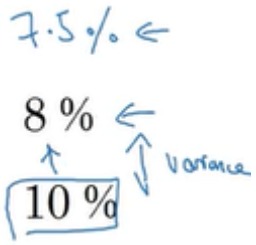
- Obtener datos etiquetados por humanos
- obtener información desde un análisis manual del error. ¿Por qué una persona hizo algo bien?
- Mejor análisis del bias/varianza.

Volviendo a la clasificación de gatos. Consideremos el siguiente ejemplo donde el conjunto de datos de entrenamiento posee un error muy grande respecto al humano.



En este sentido conviene reducir el bias. Entonces uno quiere hacer cosas como probar redes neuronales más grandes o correr conjunto de entrenamiento más tiempo.

Ahora supongamos que los humanos tienen un error del 7.5% debido a problemas de resolución, por ejemplo. Uno querría ahora centrarse en disminuir la varianza.



A la diferencia de error entre el conjunto de entrenamiento y el rendimiento humano se lo llama **Bias evitable (Avoidable Bias)**. Mientras que a la diferencia de error entre el conjunto de entrenamiento y el conjunto de desarrollo se lo llama varianza. En el segundo ejemplo el primero es de 0.5% y el segundo del 2%. Dado un indicador de la conveniencia en disminuir la varianza.

Entendimiento del rendimiento a nivel humano

Error humano como proxy del error de Bayes

Dado el siguiente ejemplo, ¿a qué consideramos como error humano?

Medical image classification example:

Suppose:

(a) Typical human ..... 3 % error

(b) Typical doctor ..... 1 % error

(c) Experienced doctor ..... 0.7 % error

(d) Team of experienced doctors .. 0.5 % error

Tal vez si se quiere demostrar que el algoritmo supera el rendimiento de un humano lo conveniente es usar (b) como error humano. Pero si el objetivo es utilizarlo como proxy para el error de Bayes entonces conviene definirlo como (d).

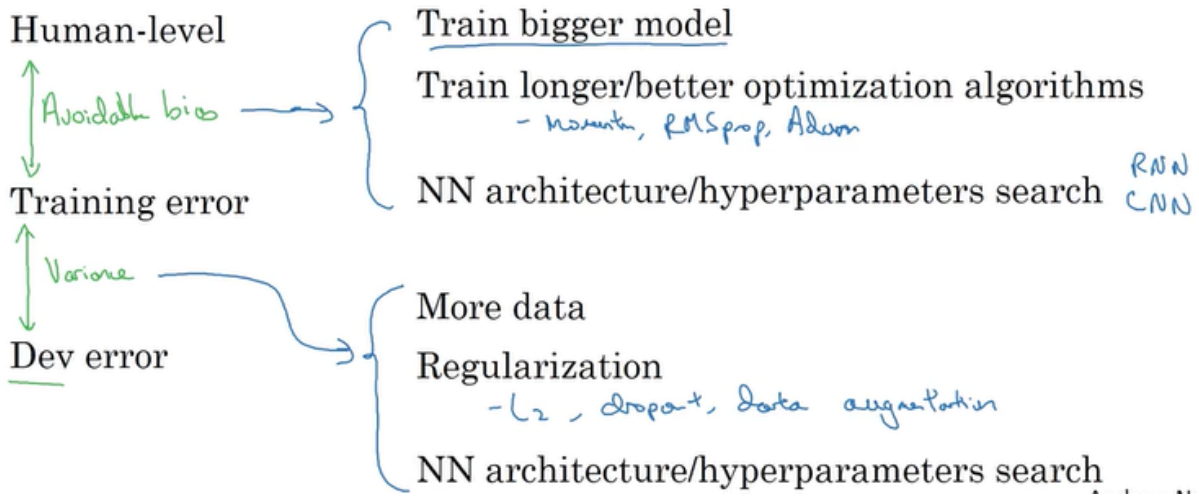
Problemas donde ML sobrepasa el rendimiento humano

Notemos que estos no son natural perception tasks como visión o NLP.

- Online advertising
- Product recommendations
- Logistics (predicting transit time)
- Loan approvals

Mejorar el rendimiento del modelo

Resumiendo lo visto.



Week 2: ML Strategy (2)

Error Analysis

Análisis de errores

Este es el proceso de examinar manualmente errores para dar una idea de cómo continuar para lograr un rendimiento cercano al humano.

Consideremos el ejemplo de categorización de perros, mirando el conjunto de datos de desarrollo obteniendo una métrica de rendimiento del 90% un error del 10%.

Uno se preguntaría si habría que mejorar la clasificación de los perros. Una solución es agarrar 100 ejemplos del dev set y contar cuantos son perros.

En el caso de que lo fuera el 5%, el error disminuiría un 0.5% solamente. Mientras que si el 50% de los errores proviene de perros el error bajaría al 5%!.

90% accuracy
→ 10% error

Should you try to make your cat classifier do better on dogs?

Error analysis:
- Get ~100 mislabeled dev set examples.
- Count up how many are dogs.

"ceiling"

5% 10%
5/100 9.5%

50% 10%
50/100 5%

Evaluar múltiples ideas en paralelo

Ideas for cat detection:

- Fix pictures of dogs being recognized as cats ←
- Fix great cats (lions, panthers, etc..) being misrecognized ←
- Improve performance on blurry images

Una idea para esto sería crear una tabla con las imágenes que se miraran como filas y las clasificaciones como columnas y se completa de acuerdo a lo que corresponda. Por último, se determina que porcentaje posee mayor error.

Image	Dog	Great Cats	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at zoo
⋮	⋮	⋮	⋮		
% of total	8%	43%	61%	12%	

Claramente muchos errores en este ejemplo provienen de imágenes borrosas o de los “grandes gatos”.

¿Qué hacemos cuando tenemos ejemplos en el conjunto de desarrollo que están mal etiquetados?

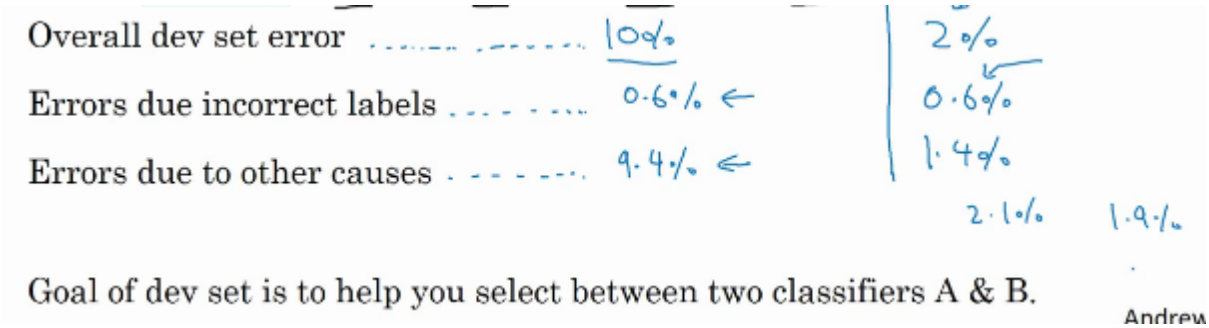
En el caso de haber ejemplos mal etiquetados en los ejemplos de entrenamiento no tiene mucho sentido preocuparse porque generalmente los algoritmos de DL suelen ser bastante robustos en los mismos.

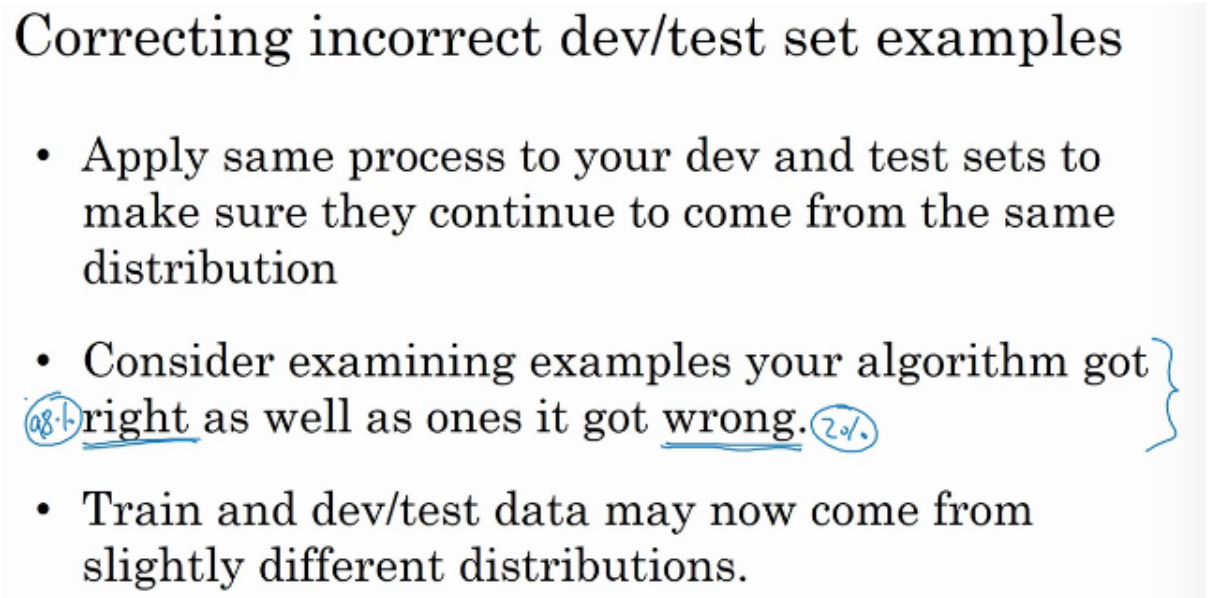
Pero, en el caso de haber en el dev/test sets se recomienda hacer un análisis de errores agregando una columna extra contando el número de ejemplos mal etiquetados.

Image	Dog	Great Cat	Blurry	Incorrectly labeled	Comments
...					
98				✓	Labeler missed cat in background
99		✓			
100				✓	Drawing of a cat; Not a real cat.
% of total	8%	43%	61%	6%	

En este ejemplo encontramos que la corrección debido al mal etiquetado no disminuiría el error significativamente. Solo sería necesario hacerlo cuando realmente valga la pena.

A veces conviene tener en cuenta otros errores asociados en el proyecto para compararlos entre si y determinar si es necesario corregir etiquetas.





Primero construye el sistema rápidamente, luego itera.

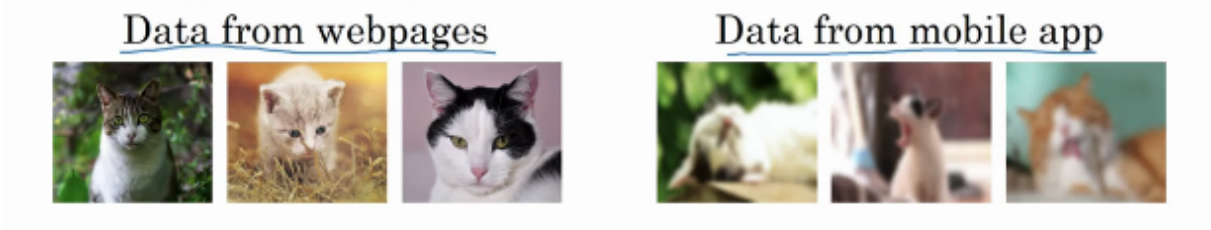
- Set up dev/test set and metric
- Build initial system quickly
- Use Bias/Variance analysis & Error analysis to prioritize next steps.

Mismatched training and dev/test set

Entrenamiento y testeo sobre distribuciones diferentes

Supongamos que una aplicación del celular pretende distinguir gatos.

En este caso se tienen dos fuentes: una proveniente de páginas web y otra de los usuarios vía la aplicación, siendo esta última menos profesional con más defectos de distorsión, enfoque, etc.

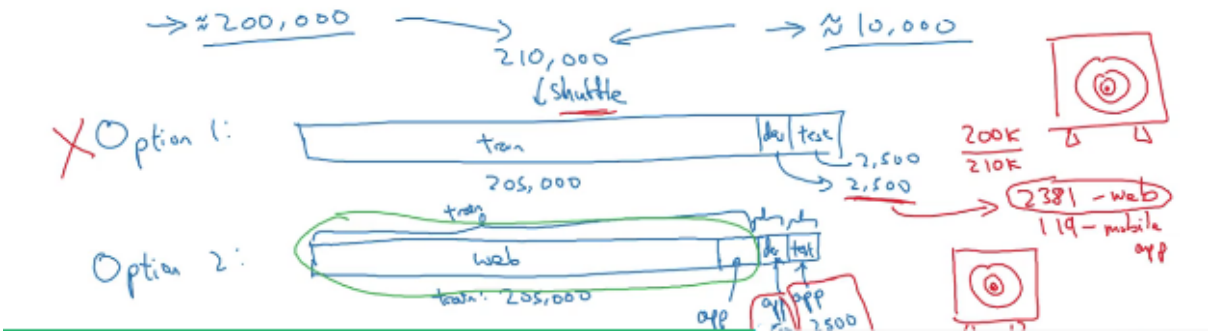


Supongamos que de la web se tienen 200mil imágenes de gatos aproximadamente y que se tiene un número relativamente bajo de usuarios, teniendo un dataset de 10mil imágenes de gatos vía aplicación móvil. Uno se encuentra en un dilema porque el sistema debería funcionar mejor sobre la distribución con menos datos.

Para resolver esto hay varias opciones.

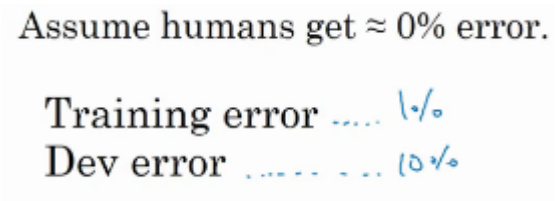
Opción 1: juntar los datos y distribuirlos de manera aleatoria. La ventaja es que la distribución será uniforme, pero posee una desventaja de que prácticamente no hay datos provenientes de la aplicación para el dev/Test set. Esta opción se recomienda no utilizar.

Opción 2: Formar el training set con las imágenes de la web completas más 5mil de la aplicación. El dev/test siendo completamente de la aplicación. Como desventaja el training set tiene una distribución distinta.



¿Es recomendable usar toda la información para la red neuronal? No.

Ejemplo clasificador de gatos.

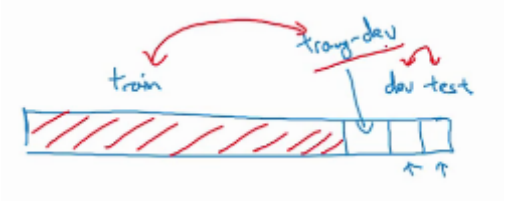


Si las distribuciones del training/dev set son las mismas, podemos concluir que habría que disminuir la varianza. Pero si provienen de distintas distribuciones esta suposición no sigue siendo válida. Tal vez no hay un error de varianza y simplemente el dev set posee imágenes más complicadas para clasificar.

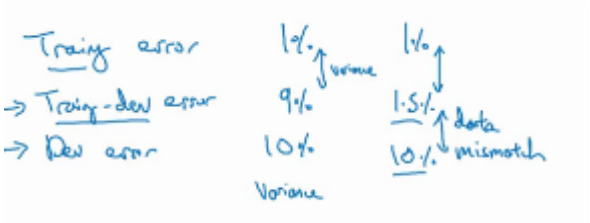
Cuando se pasó del error de entrenamiento al error de desarrollo dos cuestiones entraron en juego:

1. El algoritmo vio datos del conjunto de entrenamiento pero no de desarrollo.
2. La distribución de datos es diferente.

Para entender esto se suele definir el **Training-dev set:** tiene la misma distribución que el conjunto de entrenamiento pero no se usa para entrenar la red neuronal.

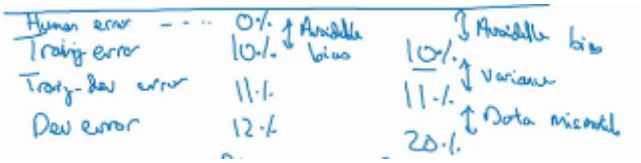


Solo se entrena el database de entrenamiento. Pongamos algunos ejemplos:



En la columna de la izquierda se puede ver que como el training y training-dev sets tienen la misma distribución el aumento del error es un problema de high variance. Mientras que en la columna de la derecha el error considerable entre el training-dev y el dev es un problema de haber entrenado la red con diferentes distribuciones, a esto se le llama data mismatch.

Otros dos ejemplos



En la primera columna se tiene un ejemplo de high bias, mientras que en segundo tanto high bias como data mismatch.

¿Qué hacer cuando se tiene un problema de data mismatch?

- Realizar un análisis de errores manual para tratar de entender la diferencia entre training y dev/Test sets.
- Hacer que los datos de entrenamiento sean más similares; o recolectar más información similar a los dev/test sets.

Datos sintetizados artificialmente

El problema con esto es que se puede estar sobre ajustando los datos sintetizados artificialmente.

Learning from multiple tasks

Transfer learning

Técnica en la que se utiliza un conocimiento aprendido en una tarea para aplicarla a otra diferente.

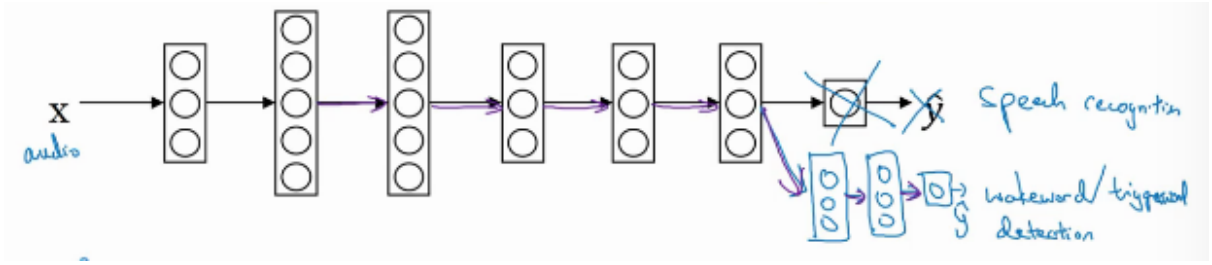
Supongamos que se entrenó una red neuronal en reconocimiento de imágenes. Si se quiere transferir esto para diagnósticos radiológicos, se puede eliminar la última capa e inicializar pesos w^{L}, b^{L} aleatorios para la última capa.

Luego se procederá a entrenar la red neuronal para esta tarea. Si el dataset es lo suficientemente grande se podría entrenar la red completa, sino la última capa únicamente.

Definiciones:

Pre-training: entrenar la red para un proceso previo.
Fine-tuning: entrenar la red para el nuevo proceso.

También se podrían agregar varias capas para el proceso de transferencia.



¿Cuándo tiene sentido este proceso? Cuando se quiere ir desde un aprendizaje con muchos datos a otro con una cantidad mucho menor.

Ej., en reconocimiento de imagen se pueden tener 1M de ejemplos, pero para radiología únicamente 100. En reconocimiento de habla 10mil horas de datos, mientras que en detección de triggereo 1 hora.

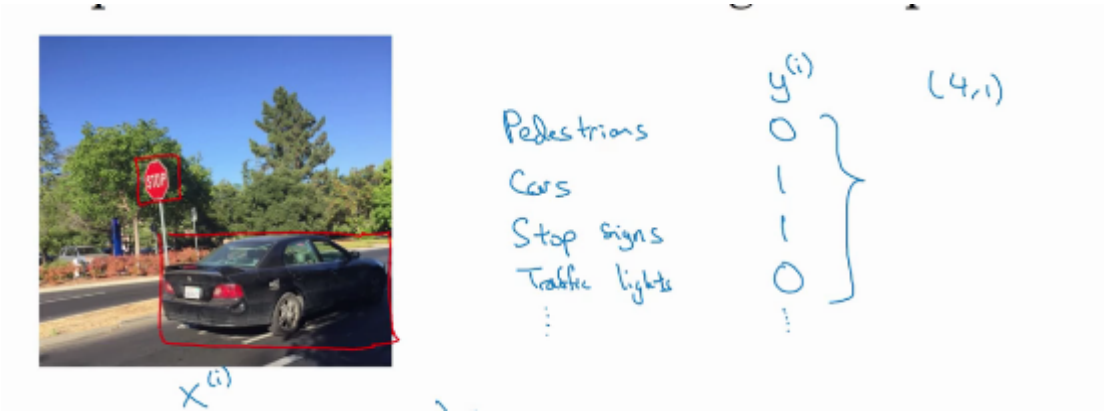
A su vez, tiene sentido cuando ambos sistemas tienen el mismo tipo de entrada, x.

Si lo opuesto fuera cierto, no sería conveniente aplicar aprendizaje por transferencia.

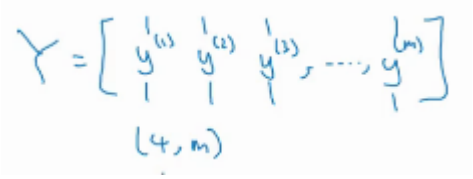
Multi-task learning

Esto es como una versión generalizada de transfer learning, aprendiendo de muchas tareas en simultáneo.

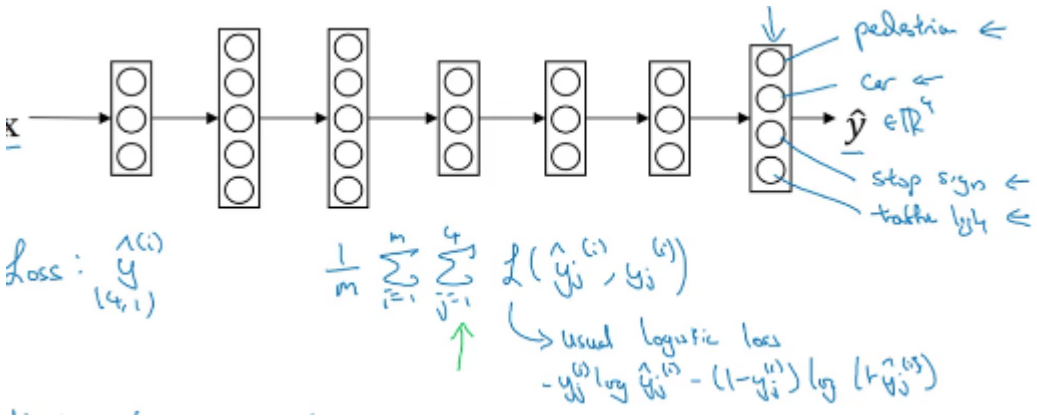
Ejemplo de conducción automática.



Mirando a los outputs como un conjunto.



Para predecir estos valores de Y.

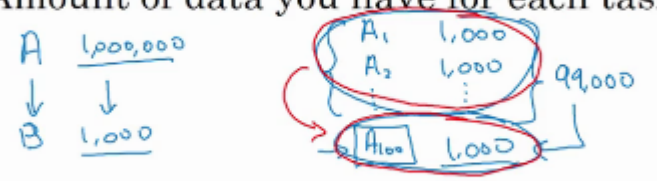


A diferencia de softmax, una imagen puede tener varias salidas. Por ejemplo, puede tener un humano y un auto.

Multitask learning es mucho más eficiente que entrenar cuatro redes neuronales para distinguir los cuatro elementos por separado.

Resumen: ¿Cuándo tiene sentido usar multitask learning?

- Training on a set of tasks that could benefit from having shared lower-level features.
- Usually: Amount of data you have for each task is quite similar.

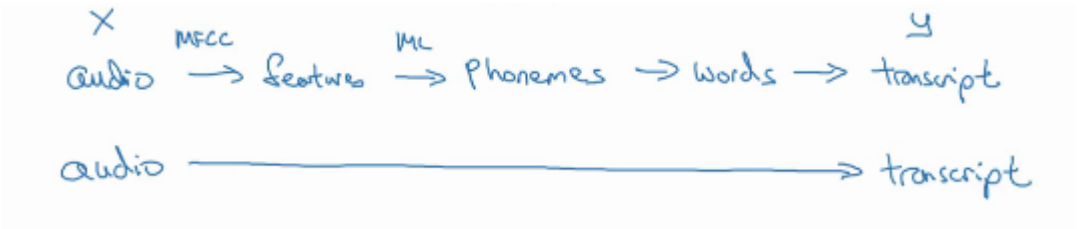


- Can train a big enough neural network to do well on all the tasks.

End-to-end deep learning

Este método toma muchas etapas de un proceso de aprendizaje y los une en una única etapa.

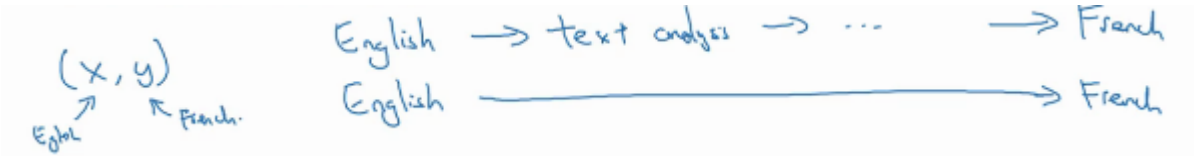
Ejemplo con reconocimiento de audio



En lugar de desarrollar piezas separadas end-to-end deep learning traduce el audio a la transcripción en un único paso.

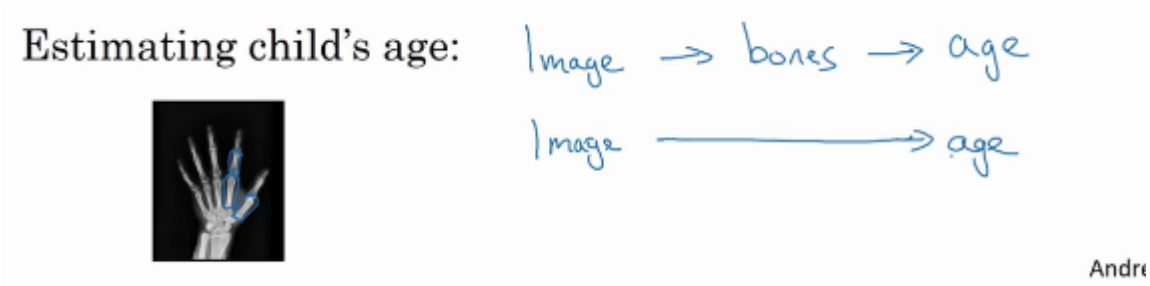
Para que esto funcione correctamente se necesitan muchos datos.

Ejemplo 2, traducciones



Ejemplo 3, estimar edad de chicos a partir de imágenes por rayos x.

Este último método no funciona del todo bien debido a la poca información de la que se dispone.



Ventajas y desventajas del end-to-end deep learning

Pros:

- Let the data speak
- Less hand-designing of components needed

Cons:

- May need large amount of data
- Excludes potentially useful hand-designed components

¿Cómo se construye un auto que se maneja solo?

Esto no es un enfoque end-to-end