1a. Linear Regression - Single Variable

```
from tensorflow import keras
from tensorflow.keras import layers as L
import numpy as np

X=np.arange(100.);y=3*X+7+np.random.randn(100)*5
m=keras.Sequential([L.Dense(1,input_shape=(1,))])
m.compile(keras.optimizers.Adam(.1),'mse')
m.fit(X,y,epochs=200,verbose=0)
print(m.predict(np.array([50.])))
```

1b. Linear Regression - Multiple Variables

```
from tensorflow import keras
from tensorflow.keras import layers as L
import numpy as np

X=np.random.rand(500,3)*10
y=2*X[:,0]+3*X[:,1]-1.5*X[:,2]+5
m=keras.Sequential([L.Dense(64,'relu',input_shape=(3,)),
  L.Dense(32,'relu'),L.Dense(1)])
m.compile('adam','mse',['mae'])
m.fit(X,y,epochs=100,validation_split=.2,verbose=0)
print(m.predict(X[:5]))
```

2a. Speech to Text

```
import speech_recognition as sr
r=sr.Recognizer()
with sr.Microphone() as s:
    r.adjust_for_ambient_noise(s); a=r.listen(s)
print(r.recognize_google(a))
```

2b. Text to Speech

```
from gtts import gTTS
import os
gTTS("Hello deep learning lab",lang='en').save("o.mp3")
os.system("start o.mp3")
```

2c. Video to Frames

```
import cv2,os
os.makedirs("frames",exist_ok=True)
c=cv2.VideoCapture("video.mp4");i=0
while True:
    ok,f=c.read()
    if not ok:break
    cv2.imwrite(f"frames/{i:04d}.jpg",f);i+=1
c.release();print(i)
```

3. FFNN - Logic Gates

```
from tensorflow import keras
from tensorflow.keras import layers as L
import numpy as np

X=np.array([[0,0],[0,1],[1,0],[1,1]],float)
G={"AND":[0,0,0,1],"OR":[0,1,1,1],"NAND":[1,1,1,0],
   "NOR":[1,0,0,0],"XOR":[0,1,1,0],"XNOR":[1,0,0,1]}
for n,y in G.items():
    m=keras.Sequential([L.Dense(8,'relu',input_shape=(2,)),
      L.Dense(1,'sigmoid')])
    m.compile(keras.optimizers.Adam(.05),'binary_crossentropy')
    m.fit(X,np.array(y),epochs=500,verbose=0)
    print(n,(m.predict(X,verbose=0)>.5).astype(int).ravel())
```

4a. Character Recognition - CNN

```
from tensorflow import keras
from tensorflow.keras import layers as L
(xt,yt),(xs,ys)=keras.datasets.mnist.load_data()
xt=xt.reshape(-1,28,28,1)/255;xs=xs.reshape(-1,28,28,1)/255
m=keras.Sequential([L.Conv2D(32,3,activation='relu',input_shape=(28,28,1)),
  L.MaxPooling2D(),L.Conv2D(64,3,activation='relu'),L.MaxPooling2D(),
  L.Flatten(),L.Dense(128,'relu'),L.Dropout(.3),L.Dense(10,'softmax')])
m.compile('adam','sparse_categorical_crossentropy',['accuracy'])
m.fit(xt,yt,epochs=5,batch_size=128,validation_split=.1)
print(m.evaluate(xs,ys))
```

4b. Character Recognition - RNN

```
from tensorflow import keras
from tensorflow.keras import layers as L
(xt,yt),(xs,ys)=keras.datasets.mnist.load_data()
xt,xs=xt/255,xs/255          # 28 steps x 28 features
m=keras.Sequential([L.SimpleRNN(128,input_shape=(28,28)),
  L.Dense(64,'relu'),L.Dense(10,'softmax')])
m.compile('adam','sparse_categorical_crossentropy',['accuracy'])
m.fit(xt,yt,epochs=5,batch_size=128,validation_split=.1)
print(m.evaluate(xs,ys))
```

5a. Image Caption - LSTM

```
import numpy as np
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input as pp
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences as pad
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Model
from tensorflow.keras.layers import *
b=InceptionV3(); ext=Model(b.input,b.layers[-2].output)
def feat(p):
    i=image.load_img(p,target_size=(299,299))
    return ext.predict(pp(np.expand_dims(image.img_to_array(i),0)),verbose=0)[0]
D={"a.jpg":"startseq a dog runs on grass endseq",
   "b.jpg":"startseq a man rides a horse endseq"}
F={k:feat(k) for k in D}
t=Tokenizer();t.fit_on_texts(D.values())
V=len(t.word_index)+1;M=max(len(c.split()) for c in D.values())
A,B,Y=[],[],[]
for k,c in D.items():
    s=t.texts_to_sequences([c])[0]
    for i in range(1,len(s)):
        A.append(F[k]);B.append(pad([s[:i]],maxlen=M)[0])
        Y.append(to_categorical(s[i],V))
A,B,Y=map(np.array,(A,B,Y))
i1=Input((2048,));f=Dense(256,'relu')(Dropout(.5)(i1))
i2=Input((M,))
w=LSTM(256)(Embedding(V,256,mask_zero=True)(i2))
o=Dense(V,'softmax')(Dense(256,'relu')(add([f,w])))
m=Model([i1,i2],o);m.compile('categorical_crossentropy','adam')
m.fit([A,B],Y,epochs=30,verbose=0)
def cap(p):
    x=feat(p).reshape(1,-1);s='startseq'
    for _ in range(M):
        q=pad(t.texts_to_sequences([s]),maxlen=M)
        w=t.index_word.get(np.argmax(m.predict([x,q],verbose=0)))
        if not w or w=='endseq':break
        s+=' '+w
    return s[9:]
print(cap("a.jpg"))
```

5b. Image Caption - CNN (no LSTM)

```
import numpy as np
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input as pp
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences as pad
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Model
from tensorflow.keras.layers import *
b=InceptionV3(); ext=Model(b.input,b.layers[-2].output)
def feat(p):
    i=image.load_img(p,target_size=(299,299))
    return ext.predict(pp(np.expand_dims(image.img_to_array(i),0)),verbose=0)[0]
D={"a.jpg":"startseq a dog runs on grass endseq",
   "b.jpg":"startseq a man rides a horse endseq"}
F={k:feat(k) for k in D}
t=Tokenizer();t.fit_on_texts(D.values())
V=len(t.word_index)+1;M=max(len(c.split()) for c in D.values())
A,B,Y=[],[],[]
for k,c in D.items():
    s=t.texts_to_sequences([c])[0]
    for i in range(1,len(s)):
        A.append(F[k]);B.append(pad([s[:i]],maxlen=M)[0])
        Y.append(to_categorical(s[i],V))
A,B,Y=map(np.array,(A,B,Y))
i1=Input((2048,));f=Dense(256,'relu')(Dropout(.5)(i1))
i2=Input((M,))
w=Dense(256,'relu')(Flatten()(Embedding(V,64)(i2)))
o=Dense(V,'softmax')(Dense(256,'relu')(concatenate([f,w])))
m=Model([i1,i2],o);m.compile('categorical_crossentropy','adam')
m.fit([A,B],Y,epochs=30,verbose=0)
# generate caption with same cap() loop as 5a
```

6a. Autoencoder - MNIST

```
from tensorflow import keras
from tensorflow.keras import layers as L
import matplotlib.pyplot as plt
(x,_),(xs,_)=keras.datasets.mnist.load_data()
x=x.reshape(-1,784)/255;xs=xs.reshape(-1,784)/255
i=keras.Input((784,));e=L.Dense(32,'relu')(L.Dense(128,'relu')(i))
o=L.Dense(784,'sigmoid')(L.Dense(128,'relu')(e))
ae=keras.Model(i,o);ae.compile('adam','binary_crossentropy')
ae.fit(x,x,epochs=20,batch_size=256,validation_data=(xs,xs))
r=ae.predict(xs[:10])
for k in range(10):
    plt.subplot(2,10,k+1);plt.imshow(xs[k].reshape(28,28),cmap='gray')
    plt.subplot(2,10,k+11);plt.imshow(r[k].reshape(28,28),cmap='gray')
plt.show()
```

6b. GAN - MNIST

```
import numpy as np,matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers as L
(x,_),_=keras.datasets.mnist.load_data()
x=(x.reshape(-1,784)-127.5)/127.5;Z=100;N=128
def blk(n,**k):return [L.Dense(n,**k),L.LeakyReLU(.2)]
G=keras.Sequential(blk(256,input_dim=Z)+blk(512)+blk(1024)
  +[L.Dense(784,activation='tanh')])
D=keras.Sequential(blk(1024,input_dim=784)+[L.Dropout(.3)]+blk(512)
  +[L.Dropout(.3)]+blk(256)+[L.Dense(1,activation='sigmoid')])
D.compile(keras.optimizers.Adam(2e-4,.5),'binary_crossentropy')
D.trainable=False
g=keras.Input((Z,));GAN=keras.Model(g,D(G(g)))
GAN.compile(keras.optimizers.Adam(2e-4,.5),'binary_crossentropy')
for ep in range(1,5001):
    r=x[np.random.randint(0,len(x),N)]
    f=G.predict(np.random.normal(0,1,(N,Z)),verbose=0)
    D.train_on_batch(np.vstack([r,f]),
      np.r_[np.ones((N,1)),np.zeros((N,1))])
    GAN.train_on_batch(np.random.normal(0,1,(N,Z)),np.ones((N,1)))
    if ep%1000==0:
        im=G.predict(np.random.normal(0,1,(16,Z)),verbose=0)*.5+.5
        for k in range(16):
            plt.subplot(4,4,k+1);plt.imshow(im[k].reshape(28,28),cmap='gray')
        plt.show()
```