به نام خدا


# تمرین سری هفتم درس یادگیری عمیق

دکتر محمدی

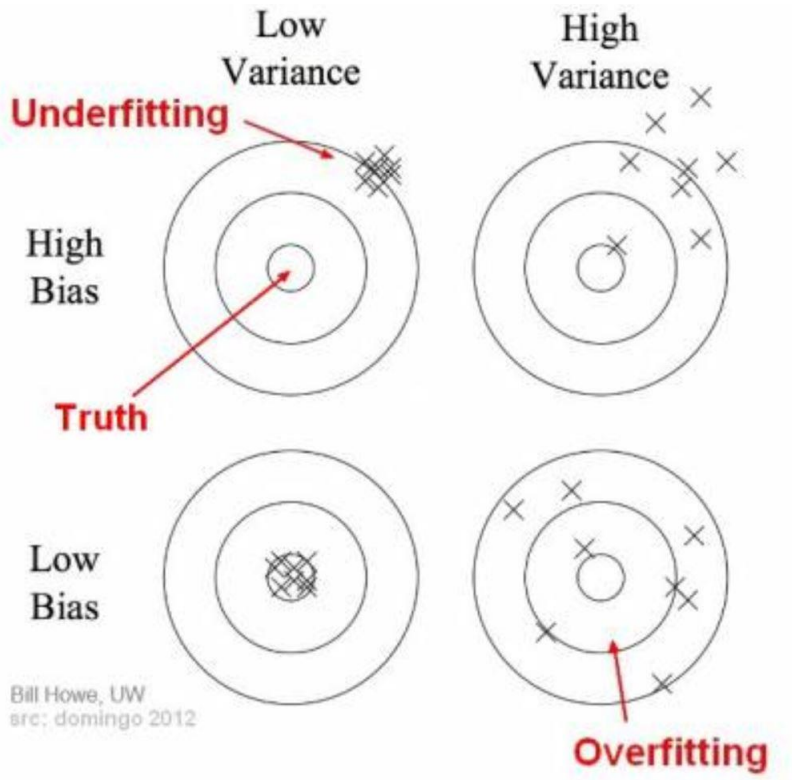
محمد یارمقدم

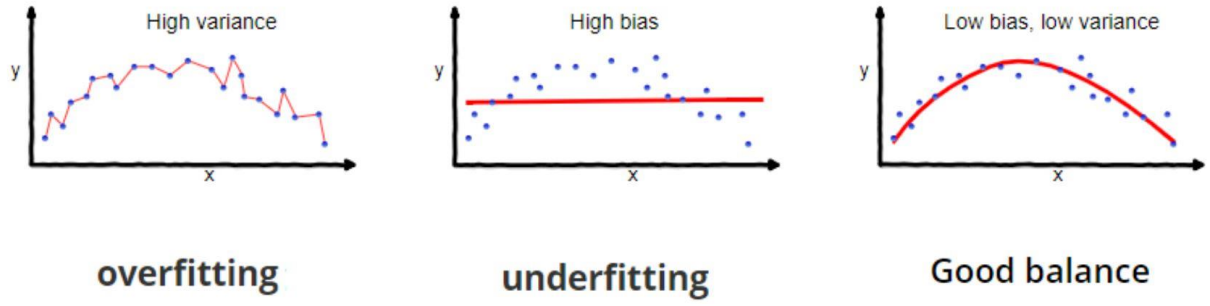
۹۶۴۶۲۱۰۴

## سوال اول)

وقتی مدل overfit شده است، نشان دهنده آن است که variance بالایی دارد و bias کمی دارد. این موضوع زمانی پیش می آید که مدل را با تعداد زیادی داده noisy آموزش دهیم. مدل با variance بالا توجه زیادی به داده های آموزش دارد و عمومیت زیادی روی داده هایی که تا کنون ندیده است ندارد. مدل با واریانس بالا نویز را احتمالا در داده های آموزش یاد گرفته است. نویز شامل نوسانات تصادفی یا جابجایی از مقادیر واقعی در ویژگی ها (متغیرهای مستقل) و پاسخ (متغیر وابسته) داده ها است. نویز می تواند رابطه واقعی بین ویژگی ها و متغیر پاسخ را پنهان کند. در این حالت مدل خیلی پیچیده میشود و در یادگیری مسائل و داده های ساد دچار مشکل می شود.

وقتی مدل underfit شده است، نشان دهنده آن است که bias بالایی دارد و variance پایینی دارد. Underfit زمانی رخ می دهد مدل که در یادگیری پترن داده های آموزش دچار مشکل میشود. این موضوع در زمانی پیش می آید که دیتای کمی برای آموزش مدل داریم. در این حالت مدل bias شده روی داده های آموزش احتمالی fit نمی شود. در این حالت مدل خیلی ساده میشود و در یادگیری مدل های پیچیده دچار مشکل می شود. این مدل های مانند linear or logistic regression هستند.



روش های حل overfit:

- Cross validation

تقسیم کردن دیتاست به بخش آموزش و تست. ساخت مدل با داده های آموزش و استفاده از داده های تست برای validation

- Regularize the weights
- Early stopping

به این معنی که چند مرحله قبل از آخرین iteration فرآیند آموزش را متوقف کنیم تا از memorizing دیتاست جلوگیری کنیم.

- Dropout

به این معنی که در فرآیند آموزش به صورت رندوم تعدادی از نورون ها را ignore کنیم.

روش های حل underfit:

- دریافت بیشتر داده های آموزشی
- افزایش سایز و تعداد پارامتر های مدل
- افزایش پیچیدگی مدل
- افزایش زمان آموزش، تا زمانی که تابع cost ، minimized شود.

**سوال دو)**

(۲)

داده های آموزش $= \begin{cases} i_1 = ۳ ,\ i_2 = ۲ ,\ o = ۸ \\ i_1 = ۱۵ ,\ i_2 = ۱۲ ,\ o = ۲۰ \end{cases}$

وزن‌های شبکه را به صورت رندوم به صورت زیر تعیین می‌کنیم:

$w_1 = -۱$ , $w_2 = ۱٫۵$ , $w_3 = -۰٫۵$ , $w_4 = ۲٫۵$ , $w_5 = ۱$ , $w_6 = ۰٫۵$

داده اول

$h_1 = w_1 i_1 + w_2 i_2 + b = -۱ \times ۳ + ۱٫۵ \times ۲ + ۰ = ۰$

$h_2 = w_3 i_1 + w_4 i_2 + b = -۰٫۵ \times ۳ + ۲٫۵ \times ۲ + ۰ = ۳٫۵ \xrightarrow{relu} ۳٫۵$

$o = w_5 h_1 + w_6 h_2 + b = ۱ \times ۰ + ۰٫۵ \times ۳٫۵ + ۰ = ۱٫۷۵ \xrightarrow{sigmoid} ۰٫۸۵۱$

داده دوم

$h_1 = w_1 i_1 + w_2 i_2 + b = -۱ \times ۱۵ + ۱٫۵ \times ۱۲ + ۰ = ۳ \xrightarrow{relu} ۳$

$h_2 = w_3 i_1 + w_4 i_2 + b = -۰٫۵ \times ۱۵ + ۲٫۵ \times ۱۲ + ۰ = ۲۲٫۵ \xrightarrow{relu} ۲۲٫۵$

$o = w_5 h_1 + w_6 h_2 + b = ۱ \times ۳ + ۰٫۵ \times ۲۲٫۵ + ۰ = ۱۴٫۲۵ \xrightarrow{sigmoid} ۰٫۹۹۹$

$\hat{y}_1 = ۸ \xrightarrow{sig} \hat{y}_1 = ۰$ $\qquad$ $\hat{y}_2 = ۲۰ \xrightarrow{sig} \hat{y}_2 = ۱$

در مرحله بعد خطا را با تابع MSE بدست می‌آوریم:

$$J = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{4}\left( \underbrace{(۰ - ۰٫۸۵۱)^2}_{۰٫۷۲۴} + \underbrace{(۱ - ۰٫۹۹۹)^2}_{۰٫۰۰۰۰۰۱} \right) = ۰٫۱۸۱$$

$$L = \frac{1}{2n} \sum_{i=1}^{n} \left( y_i - \sigma(w_5 + h_1 + w_6 h_2) \right)^2$$

$$\Rightarrow \frac{\partial L}{\partial w_6} = \frac{-1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\hat{y}_i(1 - \hat{y}_i)(h_{2,i})$$

$$\frac{\partial L}{\partial w_5} = \frac{-1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\hat{y}_i(1 - \hat{y}_i)(h_{1,i})$$

$$\frac{\partial L}{\partial w_4} = \frac{-1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\hat{y}_i(1 - \hat{y}_i)(w_{6,2})$$

$$\frac{\partial L}{\partial w_3} = \frac{-1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\, \hat{y}_i(1-\hat{y}_i)\, (w_{4,1})$$

$$\frac{\partial L}{\partial w_2} = \frac{-1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\, \hat{y}_i(1-\hat{y}_i)\, (w_{5,2})$$

$$\frac{\partial L}{\partial w_1} = \frac{-1}{n} \sum_{i=1}^{n} \underbrace{(y_i - \hat{y}_i)\, \hat{y}_i(1-\hat{y}_i)}_{f}\, (w_{5,1})$$

$$\Downarrow$$

$i=1 \longrightarrow f = (0 - 0.8801) \times 0.8801 \times 0.1569 = 0.1087$

$i=2 \longrightarrow f = (1 - 0.9999) \times 0.9999 \times 0.0001 = 0.0000001$

$$\Longrightarrow \frac{\partial L}{\partial w_4} = \frac{-1}{2}\left(0.1087 \times 0 + 10^{-7} \times 3\right) = -1.5 \times 10^{-7}$$

$$\frac{\partial L}{\partial w_5} = \frac{-1}{2}\left(0.1087 \times 3.5 + 10^{-7} \times 2.5\right) = -0.187$$

$$\frac{\partial L}{\partial w_6} = \frac{-1}{2}\left(0.1087 \times \frac{1}{2} \times 2 + 10^{-7} \times \frac{1}{2} \times 1.5\right) = -0.0543$$

$$\frac{\partial L}{\partial w_7} = \frac{-1}{2}\left(0.1087 \times \frac{1}{2} \times 3 + 10^{-7} \times \frac{1}{2} \times 1.5\right) = -0.0802$$

$$\frac{\partial L}{\partial w_8} = \frac{-1}{2}\left(0.1087 \times 2 + 10^{-7} \times 1.5\right) = -0.1087$$

$$\frac{\partial L}{\partial w_9} = \frac{-1}{2}\left(0.1087 \times 3 + 10^{-7} \times 1.5\right) = -0.1629$$

$$\Longrightarrow \frac{\partial L}{\partial w} = -\begin{bmatrix} 0.0000001 \\ 0.187 \\ 0.0543 \\ 0.0802 \\ 0.1087 \\ 0.1629 \end{bmatrix} + \delta$$

برای اضافه کردن $L_2$ regularization cost یک ترم کوچک

به این مقدار اضافه می‌کنیم:

$$\text{مومنتوم اول} = \beta_1 \times \text{moment\_f}_1 + (1-\beta) \times \left(\frac{\partial L}{\partial w} + \delta\right) \quad 0.9 \times 0 + 0.1 \times \left(\frac{\partial L}{\partial w_0} + \delta\right)$$

$$\Longrightarrow \text{unbias\_f}_1 = \frac{\text{moment\_f}_1}{1-\beta_1^{\,t}} = -\begin{bmatrix} 0.0000001 \\ 0.187 \\ 0.0543 \\ 0.0802 \\ 0.1087 \\ 0.1629 \end{bmatrix}$$

$$\text{مومنتوم دوم} = \text{moment\_f}_2 = \beta_2 \times \text{moment\_f}_2 + (1-\beta_2) \times \left(\frac{\partial L}{\partial w}\right)^2$$

$$\Rightarrow \text{unbias\_f}_2 = \frac{\text{moment\_f}_2}{(1-\beta_2^t)} = \begin{bmatrix} 0.00001 \\ 0.0015 \\ 0.0005 \\ 0.0007 \\ 0.001 \\ 0.014 \end{bmatrix}^2$$

$$w = w - \frac{\alpha}{\sqrt{\text{unbias\_f}_2}} \times \text{unbias\_f}_1 = \begin{bmatrix} -1.1 \\ 1.4 \\ -0.6 \\ 2.4 \\ 0.9 \\ 0.7 \end{bmatrix}$$

مراحل بالا را عیناً برای epoch دوم نیز تکرار می‌کنیم:

$i_1 = 3$ , $i_2 = 2 \rightarrow h_1 = 0$ , $0 = 0.08$ , $h_2 = 0$

$i_1 = 15$ , $i_2 = 12 \rightarrow h_1 = 0$ , $h_2 = 0$ , $0 = 0.0003$

$$\frac{\partial L}{\partial w} = \begin{bmatrix} 0.0014 \\ 0.002 \\ -0.006 \\ 0 \\ -0.0021 \\ 0.01 \end{bmatrix}$$

$$\Rightarrow \text{unbias\_f}_1 = \begin{bmatrix} 0.013 \\ 0.0115 \\ 0.3240 \\ -0.104 \\ 0.0112 \\ 0.00052 \end{bmatrix} , \text{unbias\_f}_2 = \begin{bmatrix} 0.043 \\ 0.00112 \\ 0.00124 \\ 0.0114 \\ 0.00134 \\ 0.0005 \end{bmatrix}$$

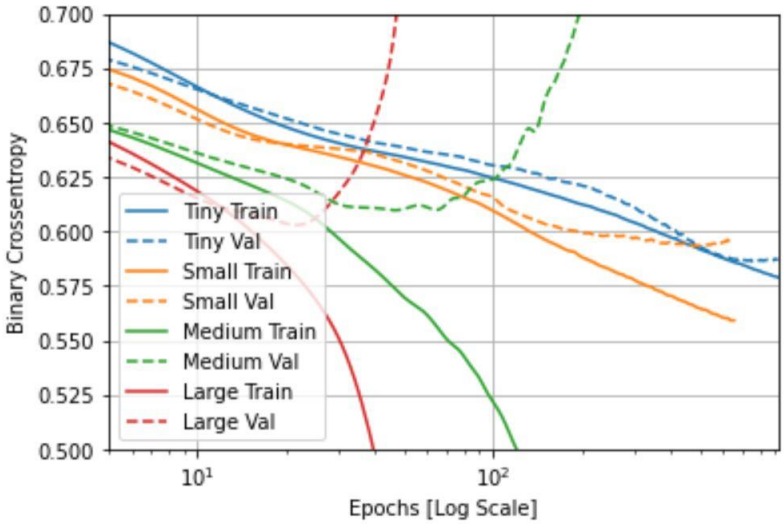
$$\Rightarrow w = \begin{bmatrix} -1.12 \\ 1.3495 \\ -0.405 \\ 2.4003 \\ 0.8009 \\ 0.402 \end{bmatrix}$$

## سوال سوم)

### الف)

اولین روش مورد استفاده در این قسمت برای جلوگیری از overfit شروع مدل با یک مدل کوچک است. این مدل در واقع مدلی که تعداد کمی پارامتر قابل یادگیری دارد. در یادگیری عمیق به تعداد این پارامتر ها ظرفیت (capacity) مدل می‌گویند. تفسیر این مورد بدین گونه است که مدل با تعداد پارامتر بیشتر ظرفیت memorization بیشتری دارد. در نتیجه روابط بین داده های آموزش و لیبل و اهداف آن‌ها را بهتر و بیشتر می‌آموزد. این مورد می‌تواند خوب باشد اما در این مورد قابلیت generalization وجود ندارد و قدرت پیش بینی در داده هایی که مدل قبلا آن‌ها را مشاهده نکرده، وجود ندارد. چالش اصلی نیز در این موارد generalization است. حال اگر مدل منابع memorization محدود داشته باشد، روابط بین داده‌ها را به راحتی نمی‌آموزد. بنابراین مجبور است روابطی که بیشترین تاثیر را در قدرت پیش‌بینی دارد بیاموزد. حال اگر مدل خیلی کوچک و محدود شود هم دچار مشکل می‌شویم و مدل در فیت شدن در داده های آموزش دچار مشکل شده و خوب آموزش نمی‌بیند. سختی کار در اینجاست که فرمول و رابطه‌ای برای یافتن تعداد پارامتر و لایه مناسب برای مدل وجود ندارد و با آزمون و خطا بدست می‌آید. بهترین روش این است که با تعداد کمی لایه و پارامتر مدل را شروع کنیم و به تدریج سایز و یا تعداد لایه ها را افزایش دهیم و این کار را تا زمانی بازده را در داده های validation در حال کاهش ببینید.

در این کد، کار با مدل tiny با دو لایه dense آغاز و سپس مدل small با اضافه شدن یک لایه dense دیگر و پس از آن مدل medium با افزایش سایز لایه های مدل از ۱۶ به ۶۴ و در نهایت مدل large با اضافه شدن یک لایه جدید و افزایش ظرفیت مدل به ۵۱۲ تشکیل شده‌اند. نتایج مدل های در نمودار زیر نمایش داده شده است.




همانطور که در نمودار مشاهده شده است فقط مدل tiny خوب عمل کرده است و در بقیه مدل ها overfit رخ داده است و هر چه مدل بزرگ‌تر شده است، overfit زودتر رخ داده است. مدل های medium و large کاملا overfit شده‌اند و مدل small هم شروع به overfit شدن نموده است. Overfit را در حالتی تشخیص می‌دهیم که خط validation loss در خلاف جهت training loss حرکت کند.

روش دوم استفاده از regularization است. از روش های مرسوم برای کنترل ovrtfit استفاده از weight regularization است. در این روش بر روی پیچیدگی شبکه محدودیت قرار می‌دهیم. به این ترتیب که شبکه را مجبور می‌کنیم تا وزن هایی با مقدار

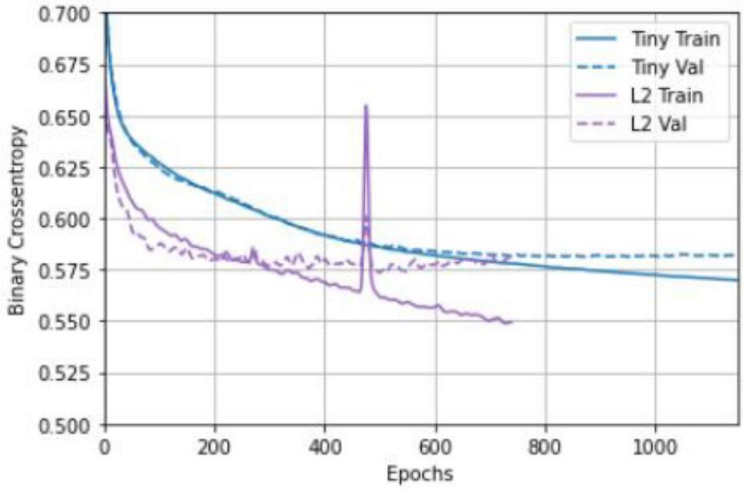
کوچک را بگیرد. با این کار distribution مقادیر وزن های شبکه منظم‌تر می‌شود. حال این کار با اضافه کردن یک هزینه که با وزن های بزرگ تعریف شده به تابع خطا شبکه انجام می‌شود. این هزینه در دو نوع تعریف می‌شود:

- L1 regularization
- L2 regularization

در نوع اول، cost اضافه شده متناسب با مقدار مطلق وزن های شبکه است.

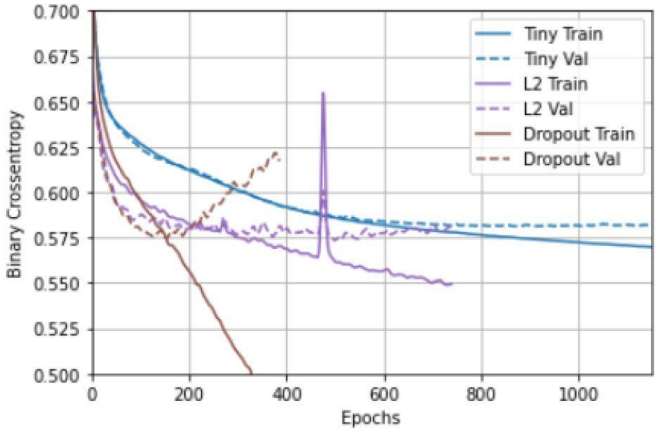
در نوع دوم، cost اضافه شده متناسب با مجذور مقدار وزن های شبکه است.

در این کد از نوع دوم برای حل مشکل overfit بهره گرفته شده است و به همه لایه ها cost به عنوان L2 Regularization اضافه شده است. اضافه شدن این ترم باعث جریمه شده وزن ها و ایجاد محدودیت در آموزش شبکه می شود.




با توجه به نمودار بالا در می‌یابیم که با وجود افزایش خطا روی داده های آموزش اما مشکل overfit در داده های large بهتر شده است، زیرا کاهش خطا در داده های آموزش باعث کاهش خطا در داده های تست نیز شده است و در یک جهت حرکت داشته‌اند.

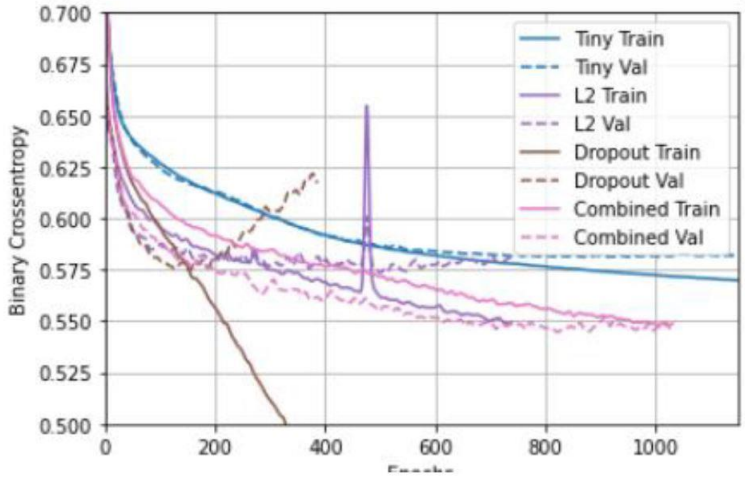
روش سوم برای حل مشکل overfit استفاده از روش dropout است. در این روش تعدادی از نورون های شبکه حذف می‌شوند. در این مثال پس هر لایه یک لایه dropout با نرخ 0.5 قرار داده شده است که باعث میشود فیت شدن شبکه به طور کامل روی داده های آموزش رخ ندهد. نتایج این روش در نمودار زیر آورده شده است



طبق نتیجه بدست‌آمده داریم که بهبود در مدل large مشاهده شده است ولی همچنان overfit وجود دارد و بهبود شرایط تا دستیابی به مدل tiny لازم است.

در روش آخر از ترکیب روش‌های بالا بهره گرفته شده است. در این مورد dropout + L2 Regularization استفاده شده است.

انتظار می‌رود که بهترین نتیجه در این مورد رخ دهد. نتیجه نمودار در زیر آورده شده است.




طبق انتظار این مورد بهترین نتیجه را به همراه داشت و از overfitting جلوگیری کرد و حتی نتیجه بهتری را نسبت به مدل کوچک به همراه داشت.

## سوال سه)

## ب)

ظبق توضیحات قسمت قبل، از ترکیب دو مورد قابل استفاده برای حل overfit استفاده کردم و تعداد لایه های گوناگونی را از تعداد کم تا زیاد تست کردم. در این مورد، دو مورد آخر به علت پیچیدگی زیاد مدل اورفیت مجدد وجود دارد و در epoch های بالا دوباره loss در داده های آموزش و تست از هم فاصله می‌گیرند و یکی روند خلاف مورد دیگر می‌گیرد.

بهترین مورد در حالت medium اتفاق افتاد که هم بهترین دقت validation و train اتفاق افتاد و هم روند دقت validation و train هر دو نزولی باقی ماند.

جزییات این مدل در ران شدن برنامه و دقت و خطای مدل در زیر آورده شده است.

```
model = tf.keras.Sequential([
    layers.Dense(128, activation='elu', kernel_regularizer=regularizers.l2(0.0001), input_shape=(FEATURES,)),
    layers.Dropout(0.5),
    layers.Dense(256, activation='elu', kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(256, activation='elu', kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(128, activation='elu', kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(1)
])

compile_and_fit(model)
```

```
 binary_crossentropy:0.6006,  loss:0.6149,  val_accuracy:0.6610,  val_binary_crossentropy:0.5836,  val_loss:0.5979,
.....
 binary_crossentropy:0.5954,  loss:0.6129,  val_accuracy:0.6740,  val_binary_crossentropy:0.5728,  val_loss:0.5902,
.....
 binary_crossentropy:0.5875,  loss:0.6069,  val_accuracy:0.6780,  val_binary_crossentropy:0.5684,  val_loss:0.5879,
.....
 binary_crossentropy:0.5879,  loss:0.6082,  val_accuracy:0.6970,  val_binary_crossentropy:0.5644,  val_loss:0.5848,
.....
 binary_crossentropy:0.5845,  loss:0.6055,  val_accuracy:0.6870,  val_binary_crossentropy:0.5616,  val_loss:0.5826,
.....
 binary_crossentropy:0.5810,  loss:0.6031,  val_accuracy:0.6750,  val_binary_crossentropy:0.5611,  val_loss:0.5832,
.....
 binary_crossentropy:0.5784,  loss:0.6014,  val_accuracy:0.6830,  val_binary_crossentropy:0.5547,  val_loss:0.5777,
.....
 binary_crossentropy:0.5740,  loss:0.5977,  val_accuracy:0.6990,  val_binary_crossentropy:0.5474,  val_loss:0.5711,
.....
  binary_crossentropy:0.5754,  loss:0.5996,  val_accuracy:0.6910,  val_binary_crossentropy:0.5465,  val_loss:0.5707,
.....
  binary_crossentropy:0.5728,  loss:0.5975,  val_accuracy:0.6860,  val_binary_crossentropy:0.5526,  val_loss:0.5773,
.....
  binary_crossentropy:0.5721,  loss:0.5972,  val_accuracy:0.6800,  val_binary_crossentropy:0.5485,  val_loss:0.5735,
.....
  binary_crossentropy:0.5715,  loss:0.5974,  val_accuracy:0.6990,  val_binary_crossentropy:0.5455,  val_loss:0.5713,
.....<keras.callbacks.History at 0x7f338d3ab690>
```

بدترین نتیجه نیز مربوط به یکی از مدل های تقریبا متوسط است زیرا بیشترین خطا را دارد و آموزش خوب رخ نداده است.

```
model = tf.keras.Sequential([
    layers.Dense(16, activation='elu', kernel_regularizer=regularizers.l2(0.001), input_shape=(FEATURES,)),
    layers.Dropout(0.2),
    layers.Dense(64, activation='elu', kernel_regularizer=regularizers.l2(0.001)),
    layers.Dropout(0.2),
    layers.Dense(64, activation='elu', kernel_regularizer=regularizers.l2(0.001)),
    layers.Dropout(0.2),
    layers.Dense(64, activation='elu', kernel_regularizer=regularizers.l2(0.001)),
    layers.Dropout(0.2),
    layers.Dense(8, activation='elu', kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(1)
])

compile_and_fit(model)
```

```
Epoch: 1300, accuracy:0.6182,  binary_crossentropy:0.6209,  loss:0.6381,  val_accuracy:0.6050,  val_binary_crossentropy:0.5989,  val_loss:0.6163,
.....................................................................................
Epoch: 1400, accuracy:0.6287,  binary_crossentropy:0.6209,  loss:0.6386,  val_accuracy:0.6280,  val_binary_crossentropy:0.5964,  val_loss:0.6143,
.....................................................................................
Epoch: 1500, accuracy:0.6349,  binary_crossentropy:0.6177,  loss:0.6362,  val_accuracy:0.6450,  val_binary_crossentropy:0.5934,  val_loss:0.6117,
.....................................................................................
Epoch: 1600, accuracy:0.6370,  binary_crossentropy:0.6150,  loss:0.6342,  val_accuracy:0.6250,  val_binary_crossentropy:0.5925,  val_loss:0.6118,
.....................................................................................
Epoch: 1700, accuracy:0.6276,  binary_crossentropy:0.6178,  loss:0.6373,  val_accuracy:0.6450,  val_binary_crossentropy:0.5875,  val_loss:0.6070,
.....................................................................................
Epoch: 1800, accuracy:0.6354,  binary_crossentropy:0.6122,  loss:0.6321,  val_accuracy:0.6430,  val_binary_crossentropy:0.5885,  val_loss:0.6084,
.....................................................................<keras.callbacks.History at 0x7f338d0d9850>
```